Hybrid Reasoning for Mobile Manipulation based on Object Knowledge

Daniel Leidner and Christoph Borst

***Abstract*— Autonomous mobile manipulation in every day environments is a key feature for next generation service robots. However, the diversity of problems that need to be solved by a robot is vast. Detailed knowledge about the environment, surrounding objects, and handling instructions is required. In this paper we combine object-centered hybrid reasoning with reachability analysis for autonomous spatial reasoning on mobile manipulation tasks. Object knowledge is used to distinguish the right place and time to move a mobile platform in order to solve a given task. The methods are evaluated in an experiment with the humanoid robot Rollin' Justin.**

I. Introduction

Hybrid reasoning, by the means of a combination of symbolic and geometric planning, is mandatory for autonomous solving of manipulation tasks. By only giving a high-level command, a robot has to reason about the symbolic order of the actions to fulfill (e.g. picking a mug before placing it in a cupboard with the available resources), but also about the geometric execution (how to pick and place in detail with the specific capabilities of the robot).

Several works have been done on symbolic and geometric reasoning to solve manipulation tasks in general. Dornhege *et al.* ground symbolic actions based on the Planning Domain Definition Language (PDDL) by adding semantic attachment modules [1]. Wolfe *et al.* directly integrate external geometric solvers into a symbolic Hierarchical Task Network (HTN) planner [2]. Karlsson *et al.* [3] invoke geometric actions during the symbolic planning phase and propose geometric backtracking in case of failure. Kaelbling *et al.* focus their work on symbolic and geometric planning under uncertainty for mobile manipulation [4]. However, when mobility is required to solve a task it is necessary to additionally reason about two questions:

- Where should a mobile robot move to fulfill a task?
- When does a mobile robot has to move its mobile base?

The topology (*"where"*) of a manipulation scenario is crucial for the success of a task. Depending on the position of a robot, objects are possibly out of reach. Manipulating such an object is thus unfeasible. Zacharias *et al.* propose to use *capability maps* [5] to represent the workspace for robotic manipulators. Capability maps can be used to position the mobile base best possible to manipulate objects or follow workspace trajectories [6]. Stulp *et al.* use capability maps to guide the positioning of a mobile base to so-called *Action-Related Places* to manipulate objects under uncertainty w. r. t. the performed action [7]. *Semantic maps*, as generated from sensor input by Nüchter *et al.* [8] are used by Galindo *et al.* in combination with symbolic planning to command mobile robots to reach a desired goal location [9]. The second question for the chronology of mobile manipulation (*"when"*) is often ignored and simply solved by scheduling additional operations that navigate to a certain goal. In this paper we want to address both questions by directly integrating a spatial reasoning step into the hybrid planning procedure.

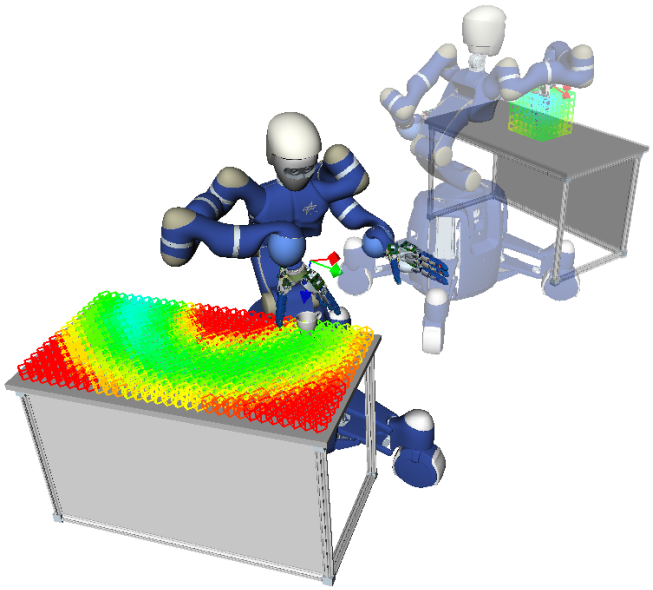


Fig. 1. A mobile pick-and-place task with the humanoid robot Rollin' Justin. Prior object knowledge is used to determine the optimal position to place the mobile base of the robot.

The so far listed methods disregard an appropriate knowledge representation to store the information needed to solve the wide variety of household tasks. Tenorth *et al.* [10] rely on a web database to exchange knowledge about actions, environments and objects between different robots. A related approach tries to parse human readable knowledge bases in the Internet to solve manipulation tasks [11]. An interesting approach is to arrange the required knowledge around the object itself. Kallmann *et al.* [12] store articulation trajectories within the objects while Levison [13] classifies objects by functionality and augments the symbolic domain with hierarchical action properties.

In our previous work we introduced a modular architecture to solve manipulation tasks from the object point of view rather than from the robot point of view. We argued that all

All authors are affiliated with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany, daniel.leidner@dlr.de

information needed to solve manipulation tasks can be stored within the descriptions of the involved objects [14]. A two step hybrid reasoning approach based on object knowledge is used to solve manipulation tasks. Our contribution in this paper is the extension of this architecture towards mobile manipulation. Object knowledge is used to distinguish the optimal positioning of a robotic mobile base w. r. t. the actions to be performed. The decision to move a mobile base or not is directly integrated into the hybrid planning procedure.

The paper is organized as follows: After an introduction to our preliminary work on hybrid reasoning in Sec. II, we extend the architecture towards mobile manipulation in Sec. III. The approach is validated in an experiment in Sec. IV. An extended discussion on the concept is provided in Sec. V.

II. Preliminary Work

Autonomous solving of everyday manipulations tasks relies on various abilities. A robot needs to localize the relevant objects in its environment, it has to plan feasible collision-free motions to manipulate them, and it has to navigate through unstructured human environments. Even though different robots have different implementations for these modules, the basic procedure to solve a certain task stays the same. For this reason it is beneficial to develop manipulation tasks in a reusable way. A robot-independent modular architecture to solve manipulation tasks is therefore essential. Focusing on special robot abilities to solve a task, on the contrary, is not desirable.

In our previous work [14] we proposed to categorize objects in a hierarchical structure according to their functionality and additionally store process models which define arbitrary manipulation instructions. We argued that all information needed to solve manipulation tasks can be stored within the descriptions of the involved objects. A hybrid planning approach is used to solve the given tasks on the symbolic as well as on the geometric level. The underlying architecture is reviewed in this section.

Our approach is based on the integration of object knowledge within the hybrid reasoning procedure. An *object storage* provides knowledge for all available objects. The objects are categorized by functionality and hierarchically arranged in the object-oriented paradigm. Objects of the same object class can be manipulated the same way since they share the same process models to handle them. However, the object oriented paradigm allows the consideration of their specific properties such as size and shape. The actual belief state for the environment of the robot is instantiated in the *world representation*. Objects as described in the object storage are handled here with specific symbolic and geometric properties.

The main component for solving manipulations tasks within the object context is formed by the so-called *action templates*. They can be described in two segments as illustrated in the center of Fig. 2. The first segment provides symbolic action definitions for symbolic planning.

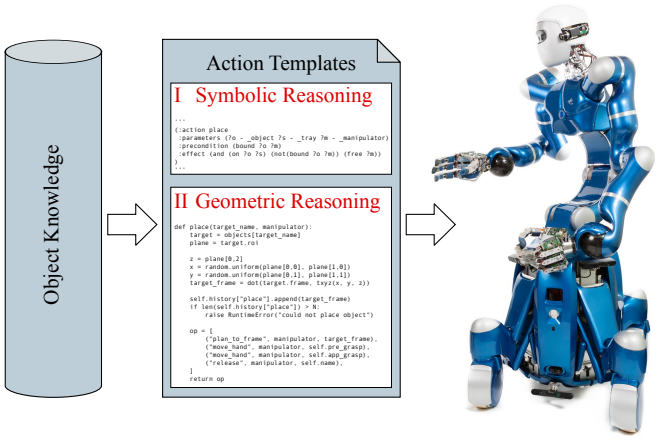
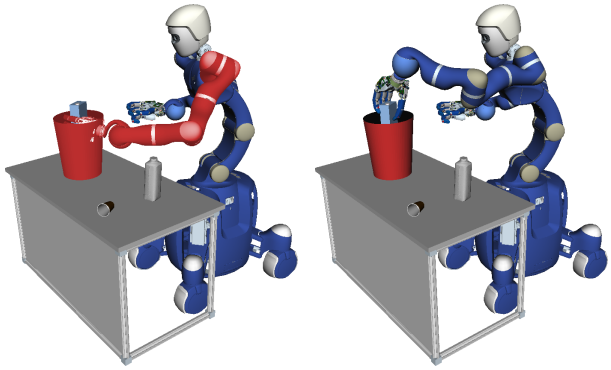


Fig. 2. Object knowledge is used to parameterize the hybrid reasoning procedure on the symbolic and geometric level. Action templates are interpreted in a two step approach. First, the symbolic action template header is parsed to solve a given task with symbolic reasoning. Second, the resulting symbolic transition is grounded based on the geometric process model, given by the second part of the action template. Geometric backtracking is inherent in this step. Finally, the feasible solution is executed on the real robot.

The second segment is executed at run-time. It defines the *symbol grounding* to the geometric level by the use of object knowledge. The procedure for manipulating objects by the use of action templates as illustrated in Fig. 2 is described as follows.

The first step to solve a manipulation task is the symbolic reasoning. The symbolic domain is generated out of the current belief state stored in the world representation. The action templates define symbolic action definitions with symbolic preconditions and effects. Additionally every object type provides symbolic predicates. With this information a symbolic planner is able to generate feasible symbolic transitions leading to a user-specified goal state.

In a follow-up step, the second part of the action templates is revisited to solve the task on the geometric level. Geometric simulation modules are therefore integrated to ground the symbolic transition. Object knowledge is heavily used to describe the process models for the different actions. Among others, CAD data, grasp sets and regions of interest (e.g. table planes) are used to describe simple pick-and-place actions. Tool frames and task trajectories are used to describe more complex actions such as tool handling. If one action is successfully simulated, the next actions are performed until the goal state is reached. Should one step of the geometric simulation fail, the action template is reviewed for geometric alternatives. After all alternatives have been attempted without success, geometric backtracking is initiated to find a previously scheduled action with remaining alternatives and the procedure starts over. If the symbolic transition is not feasible with any of the given geometric alternatives, the complete procedure is re-initialized with a different symbolic transition. Fig. 3 illustrates the backtracking mechanism in a pick-and-place example where a milk box has to be placed in a trash can.



(a) (b)

Fig. 3. The backtracking mechanism illustrated in a pick-and-place example. A side grasp is not suitable to place the milk box in the trash can due to collisions (a). After backtracking to the previous pick action, a top grasp is chosen to pick the milk box so that putting it down becomes feasible (b).

Our previous work showed that complex manipulation tasks such as using mechanisms or bi-manual tools can be solved with this architecture. Manipulating articulated objects with two arms was shown in an example where Justin had to use a hedge shear to cut a ribbon [14]. The object-oriented paradigm is capable of polymorphism, so that requesting the same action from several objects results in different instruction sets. For example serving ketchup with a bottle or a dispenser has the same effects to the world, but the way of handling the objects is very different.

The process model for the object handling as well as the backtracking mechanism are directly mapped to the action templates. Therefore, the programmer gets a different view on the problem which results in a different way of solving the task. Rather to think about the capabilities of a robot, the functionality of the objects gets into focus.

III. Extension towards Mobile Manipulation

Mobility is crucial for service robots in human households. Objects in human environments are widely spread over different rooms and different storage positions. Even simple fetch-and-carry tasks require a robot to navigate from one object to another. The main issue to move towards an object is because it is out of the reach of the robot. However, finding a suitable base position to manipulate a certain object is not trivial. The limiting factors are collisions with obstacles, limitations of the end effector kinematics and the reachability of the robotic manipulator.

For now, only a motion planner for robotic manipulators was used as geometric simulation in the proposed architecture. One could simply extend the motion planner by the Degrees of Freedom of a mobile base to solve mobile manipulation tasks as shown by Schulman *et al.* [15]. However, if the goal position to manipulate an object is ambiguous, as it is for tasks like placing a mug on a shelf, this approach can not be applied directly. Therefore, we take advantage of precalculated reachability information to find the most flexible base position in terms of reachability for the involved objects.

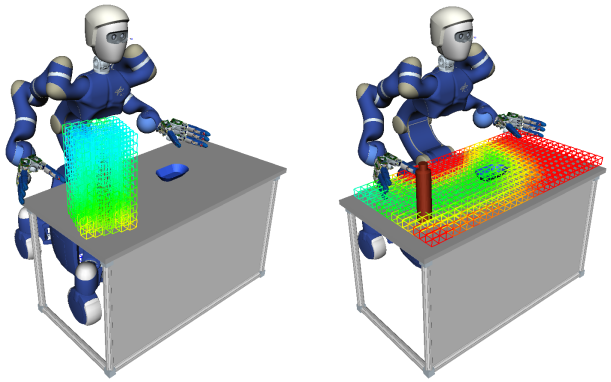
Zacharias *et al.* use so-called *capability maps* [5] to represent the reachability for robotic manipulators. It was shown that reaching for objects [7] as well as following workspace trajectories [6] can be guided by the use of this representation. We expand these approaches to cover whole *Regions of Interest (ROI)* in the workspace of the manipulator. The regions are gathered out of the object database and vary depending on the type of object. Small objects a robot can manipulate with one manipulator (e.g. mugs and bottles) form a ROI based on the dimension of the object bounding box. Bigger objects define a region of interest according to their purpose. For tables and shelves the regions of interest are defined by the dimension of the storage elements.

We optimize the mean reachability value r_{roi} in these regions by adapting the base position $\boldsymbol{p}_{\text{base}}$ and the configuration of the torso $\boldsymbol{q}_{\text{torso}}$, where $\boldsymbol{x} = (\boldsymbol{p}_{\text{base}}^T, \boldsymbol{q}_{\text{torso}}^T)^T$.

$$\boldsymbol{x}_{\max} = \arg\max_{\boldsymbol{x}} \; r_{\text{roi}}(\boldsymbol{x}) \tag{1}$$

By maximizing the reachability within the region of interest, an optimal solution vector $\boldsymbol{x}_{\max}$ is obtained. The initial position for optimizing the base is set to the object position. The joint values for the initial torso configuration are set to be upright. The optimal intersection between the capability map of the right manipulator and the ROI of the task related objects is shown in Fig. 4. The colors indicate the reachability r for the discrete voxels. Red voxels are unreachable ($r = 0.0$) and blue voxels are fully reachable ($r = 1.0$). The maximum reachability $r_{\max}$ for the manipulators of the humanoid robot Rollin' Justin is 0.833. The *inverse reachability* approach by Vahrenkamp *et al.* [16] to find suitable base positions for a particular 6D target position is comparable to our method. However, it is hardly applicable to optimize the overall reachability within an arbitrary target region.

An A* planner [17] is used to navigate to the closest possible collision-free target position as described in algorithm III.1. Depending on the object the appropriate ROI is



(a) (b)

Fig. 4. Optimal collision-free intersection of the capability map for the right manipulator and the ROI of a ketchup bottle on the left ($r_{\text{roi}} = 0.66$), and the optimal intersection for the ROI of a table on the right ($r_{\text{roi}} = 0.32$).

selected. The A* planner searches for the shortest path to reach this goal. However, the robot cannot reach the exact goal position since it is blocked by the target object itself. The map is thus initially empty and the planning algorithm is used to explore the map. Therefore, the path has to be checked for collisions afterwards. In case of collision the map position gets marked as occupied. If the robot is yet close enough to reach the object ($r_{\text{roi}} \geq 0.1$), the final position is optimized as shown in (1). The procedure is repeated until a feasible path is found or the complete map is explored without success.

Algorithm III.1: NAVIGATETOOBJECT(*object*)

$map \leftarrow$ CREATEEMPTYMAP()
$reached \leftarrow False$
$roi \leftarrow object.roi$

while $reached \neq True$
do
- $path \leftarrow$ ASTARFINDPATH(map, roi)
- **if** CHECKCOLLISION$(path) = False$
  - **then** $\{reached \leftarrow True$
  - **else**
    - UPDATEMAP(map)
    - **if** CHECKREACHABILITY$(roi) \geq 0.1$
      - **then** $\{reached \leftarrow True$

OPTIMIZEREACHABILITY(roi)

However, the solution so far provides only the information about the position *"where"* a robot should move. It is not yet specified *"when"* the platform should move or when that is not efficient. Depending on the initial reachability it is maybe better to stay still since the object is anyway in reach of the manipulator. Moving the base could corrupt the environmental belief state due to uncertainties in the odometry. Furthermore, navigating with a mobile base is highly time-consumptive due to planning and execution of the motion and necessary world state updates. For both reasons a mobile base should thus only be used when it is really necessary to solve a task.

A simple way to integrate mobility on the symbolic reasoning level is to augment the PDDL action definitions by additional preconditions for reachability validation. The PDDL description for the generic pick action would hereby look as follows:

```
:action _object.pick
  :parameters (?o - _object ?t - _tray ?m - _manipulator)

  :precondition (and (free ?m)
                     (on ?o ?t)
                     (reachable ?o ?m))

  :effect (and (bound ?o ?m)
               (not (free ?m))
               (not (on ?o ?t)))
```

The additional `(reachable ?o ?m)` predicate in the precondition section will force a robot to navigate to a certain object in every symbolic step. However, there is no geometric feedback in pure symbolic planning to reason about this decision. Additionally the generality of the symbolic domain decreases since the extra precondition has to be considered for every action. Using hybrid reasoning, this decision can be postponed to the geometric reasoning step. Instead of directly executing an action template, the reachability for the action-related object is first validated w. r. t. the object ROI. The *NavigateToObject* function as described in algorithm III.1 is executed when the reachability is beneath a minimal reachability threshold of $r_{\text{min}} = r_{\text{max}} * 0.5$, which means that the object must be at least 50% reachable by the robotic manipulator. After the object is within reach, the action template is parsed and executed. No symbolic overhead is generated this way and a seamless integration of the navigation module within the hybrid reasoning procedure is thereby achieved.

IV. EVALUATION

The evaluation of the proposed methods is done by examining the capabilities of the overall architecture in an experiment for mobile manipulation. The experiment is carried out in simulation with the mobile humanoid robot Rollin' Justin of DLR [18]. The mobile base of the robot allows for omnidirectional motions. The expandable torso allows to reach for objects at different heights. Both abilities are exploited to optimize the reachability of the manipulators w. r. t. the provided object knowledge. The sequence of the experiment is illustrated in Fig. 5.

The task to be solved is a pick-and-place scenario, in which two mugs have to be placed on a shelf. The objects are too far away from each other to reach them at the same time. The robot is forced to use the mobile base to fulfill the task successfully. The first step is to approach the mugs from distance, so that they can be picked up. The second step is to transport them to the shelf and place them down. The position to put the mugs down is arbitrary. It is defined by the ROI of the shelf, respectively the storage plane. The symbolic transition given by the symbolic planner looks as follows:

```
_object.pick mug1 table1 right_arm,
_object.place mug1 shelf,

_object.pick mug2 table2 right_arm,
_object.place mug2 shelf
```

The reasoning about the navigation is done inherently and does not appear in the symbolic transition. As seen in Fig. 5 the object-specific regions of interest are used to determine the optimal positioning of the mobile base w. r. t. the right manipulator, while collisions with the environment are taken into account. The A* planner is used to determine the base trajectory from one point to another. The final base position and torso configuration are optimized according to (1).

The experiment shows that mobile pick-and-place tasks are solvable with the proposed architecture. The robot autonomously decided to navigate in between the single actions by only using the provided object knowledge. The placement of the mobile base as well as the configuration of the torso was chosen optimal, w. r. t. the executed actions and the involved objects. Experiments with more realistic scenarios and further evaluation on the real robot are scheduled for future work.

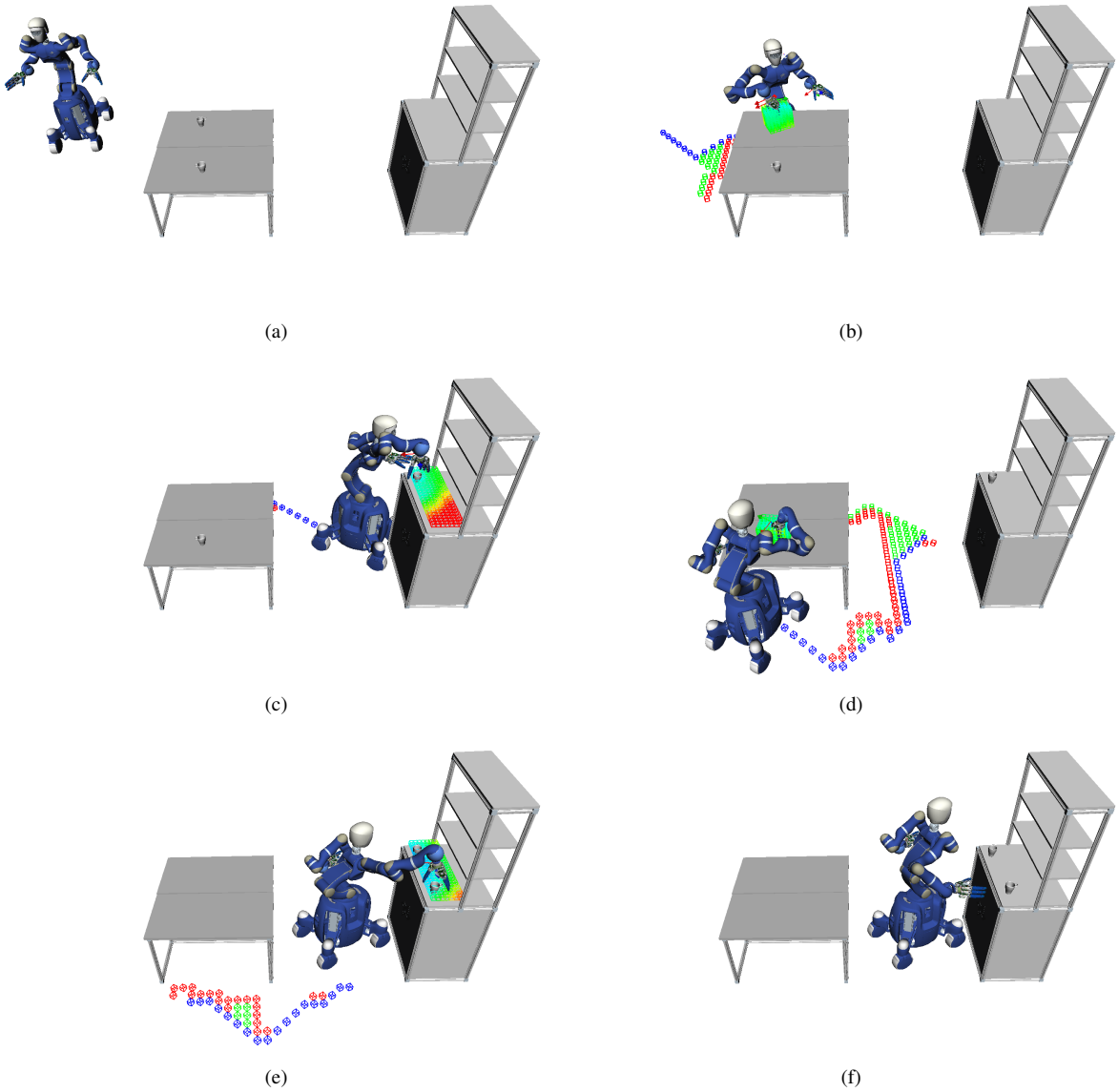


Fig. 5. The two mugs have to be carried to the shelf. The robot has to navigate towards the mugs, pick them up, and place them on the shelf. The symbolic transition does not include the navigation. The need for moving the mobile base is autonomously decided w. r. t. the reachability as provided by the capability map. The A* path is illustrated as colored blocks on the floor where red blocks are occupied, green ones are free, and the blue blocks indicate the final path.

V. Discussion

The proposed concept is expressive enough to easily describe various manipulation tasks. However, some drawbacks have been discovered while developing robotic manipulation scenarios. The advantages and limitations of the architecture are discussed within this section.

We found that the main benefit of our approach is the way of programming applications. Since the focus is on the object and the related action rather than on a special robot, different aspects of the problem are examined. It is quite natural to define workspace trajectories in the coordinates of an object whereas end effector trajectories for a robot manipulator are impractical to describe a task since they are often not aligned with the tool. The same applies to regions of interest as shown in this paper. Therefore, we enforce to store task trajectories and tool frames within object databases independent of specific robot capabilities.

Furthermore, the integrated backtracking mechanism within the action templates is very intuitive. Since the programmer already has a detailed understanding of the process model of the action currently under development, the next logic step is to think about possible situations leading to failure. Having this in mind, suitable failure recovery originates in a natural way. It is even possible to react on previous attempts and change the behavior for the next approach accordingly.

As shown in this paper, another advantage is the simple adaption of the architecture to different problems. The modular geometric planning environment allows to include special

modules for any kind of problem. For the future we plan to integrate modules for adapting low-level control behaviors, reacting on environmental effects, as well as validation and verification of ongoing executions.

However, there are some limitations yet. One issue with hybrid planning is the time consuming reasoning process. The geometric evaluation of the symbolic transition is the bottle-neck. Due to a large number of possible alternatives, the search for right combinations can take up to several minutes depending on the complexity of the problem. The backtracking as described in Sec. II could be designed more efficient, since the alternatives are simply chosen in a predefined order (e.g. the order of the grasps stored in the object database). One way to overcome this issue would be to use the alternatives based on a machine learning approach on similar problems solved in the past. However, deciding which problems are related to each other is not straightforward. They may vary in symbolic aspects (e.g. placing two bottles rather than two mugs on a tray) and geometric aspects (e.g. the topology of the problem) but describe basically the same situation. Suitable clustering criteria have to be defined.

During our work on mobile manipulation we recognized an additional drawback regarding the two step hybrid reasoning approach. We found that the symbolic planner may output sub-optimal solutions to solve a task due to the lack of geometric information. Looking at the example in Sec. IV, it is obvious that the robot could solve the task more efficient when both manipulators are used at the same time to transport the mugs to the shelf. One could solve this issue by integrating geometric cost computation during the symbolic reasoning step. A rudimentary geometric representation is thereby sufficient, so that the resulting extra computation time does not slow down the overall planning process too much.

VI. Summary

In this paper we presented a generic hybrid reasoning approach for mobile manipulation. Our previous work has been extended by reasoning modules for reachability and navigation. The optimal base position as well as the corresponding torso configuration is chosen based on the information of the objects involved in the task and the reachability of the robotic manipulator w. r. t. the capability map. Spatial reasoning is used to autonomously initiate navigation towards the related objects between the single actions. A simulated experiment evaluates our approach.

References

[1] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *Towards Service Robots for Everyday Environments*. Springer, 2012, pp. 99–115.

[2] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2010.

[3] L. Karlsson, J. Bidot, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[5] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2007, pp. 3229–3236.

[6] F. Zacharias, W. Sepp, C. Borst, and G. Hirzinger, "Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories," in *IEEE/RAS International Conference on Humanoid Robots (ICHR)*, 2009.

[7] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz, "Learning and reasoning with action-related places for robust mobile manipulation," *Journal of Artificial Intelligence Research*, vol. 43, no. 1, pp. 1–42, 2012.

[8] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.

[9] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008.

[10] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "The roboearth language: Representing and exchanging knowledge about actions, objects, and environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[11] M. Beetz, U. Klank, A. Maldonado, D. Pangercic, and T. Rühr, "Robotic roommates making pancakes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] M. Kallmann and D. Thalmann, "Modeling objects for interaction tasks," in *Eurographics Workshop on Computer Animation and Simulation*, 1998.

[13] L. Levison, "Connecting planning and acting via object-specific reasoning," Ph.D. dissertation, University of Pennsylvania, 1996.

[14] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *IEEE/RAS International Conference on Humanoid Robots (ICHR)*, 2012.

[15] J. Schulman, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems Conference (RSS)*, 2013.

[16] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[18] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, *et al.*, "Rollin'justin-mobile platform with variable base," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.