Robot Task Planning with Contingencies for Run-time Sensing

Andre Gaschler, Ronald P. A. Petrick, Torsten Kröger, Alois Knoll and Oussama Khatib

Abstract— **In this work, we present a general approach to task planning based on contingent planning and run-time sensing, which forms part of a robot task planning framework called KVP. Using the general-purpose PKS planner, we model information-gathering actions at plan time that have multiple possible outcomes at run time. As a result, perception and sensing arise as necessary preconditions for manipulation, rather than being hard-coded as a task itself. We demonstrate the effectiveness of our approach on two simple scenarios covering visual and force sensing, and discuss its applicability to more general tasks in automation and mobile manipulation, involving arbitrary numbers of sensors and manipulators.**

I. Introduction

In order to model realistic environments for robot task planning, symbolic task planners need to reason about incomplete knowledge and perceptual information as provided by sensors. In order to facilitate this task and apply general purpose planning to the robotics domain, we developed the ***K**nowledge of **V**olumes framework for robot task **P**lanning (KVP)*, which was initially presented in [1].

KVP is guided by three principles, which make it useful for a broad range of robot task planning applications that require incomplete knowledge, real-world geometry, and multiple robots and sensors: **(1)** KVP uses PKS (Planning with Knowledge and Sensing) [2], [3], a general-purpose symbolic planner that operates at the knowledge level. PKS can represent known and unknown information, and model sensing actions using clear and concise domain descriptions, making it well suited for reasoning in structured, partially known environments of the kind that arise in many robot scenarios. **(2)** KVP is one of the first approaches to treat 3D geometric volumes as an intermediary representation between continuously-valued robot motions and discrete symbolic actions, tackling the general problem in robot task planning of bridging the gap between geometric and symbolic representations. **(3)** By using the intermediate representation of volumes, KVP can model continuous geometry, in contrast to arbitrary discretization, as discussed in [1].

The central contribution of this work is to apply general-purpose, contingent planning techniques to the robotics domain and demonstrate the effectiveness of this approach in two scenarios, namely a Force Sensing (Fig. 1) and a Visual Sensing (Fig. 3) scenario.

A. Gaschler is with fortiss GmbH, affiliated with the Technische Universität München, Munich, Germany. Correspondence should be addressed to gaschler@fortiss.org

R. Petrick is with the School of Informatics, University of Edinburgh, Edinburgh, United Kingdom.

T. Kröger and O. Khatib are with the Artificial Intelligence Laboratory, Stanford University, Stanford, USA.

A. Knoll is with the Department of Informatics, Technische Universität München, Munich, Germany.

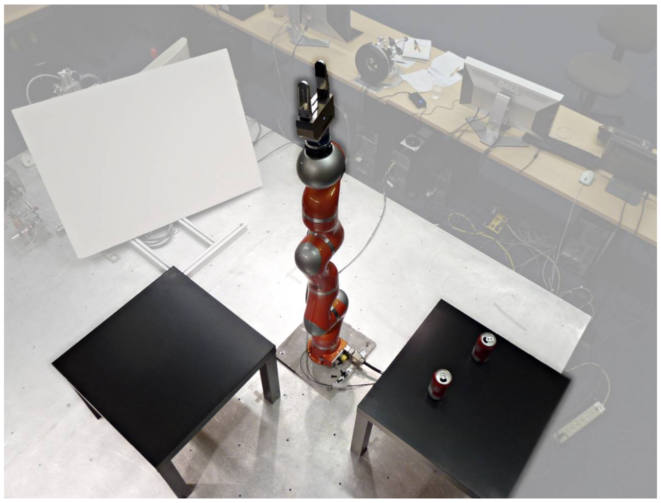

Fig. 1. In the Force Sensing scenario, a compliant robot manipulator can sense if beverage containers are filled by weighing them, and holding them upright while moving to prevent spilling, unless they are known to be completely empty or not opened.

In the following, we first compare our approach with existing solutions. We then discuss our framework in Sec. II and demonstrate its effectivess in two scenarios in Sec. III. Finally, we discuss future work and conclude in Sec. IV.

A. Related Work

Early work on robot task planning dates back to systems like Shakey [4] and Handey [5]. Since that time, the field has made significant developments, and the general problem of robot task planning has been approached from diverse areas of research, including probabilistic techniques from artificial intelligence [6], closed-world symbolic planning [7], [8], [9], formal synthesis [10], [11], and manipulation planning [12].

A recent and notable contribution is the belief space planner by Kaelbling and Lozano-Pérez [6], which models a belief space of probability distributions over states, making it robust against uncertainty and change. In contrast to belief states, our work instead relies on discrete knowledge and is designed for structured environments with incomplete information and sensing. Furthermore, while Kaelbling and Lozano-Pérez use octrees to represent swept volumes of robot motion, we use sets of convex shapes, allowing efficient collision detection in the deterministic case [1]. In both cases, sensing actions are formulated as preconditions for manipulation, rather than being hard-coded as tasks themselves.

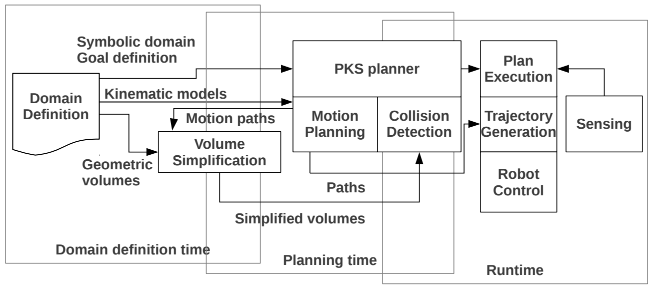

Fig. 2. Overview of the implemented KVP software architecture [1].

A number of approaches also address the problem of integrating symbolic planning and motion planning. For instance, our work is in part inspired by Kaelbling and Lozano-Pérez's earlier work on hierarchical task and motion planning in [13], borrowing the continuous geometry of swept volumes. However, while the geometric preconditions may be similar, their underlying aggressively hierarchical planning strategy differs from the knowledge-based planner we use here, which has also been used in prior work to connect robot vision and grasping with symbolic action [14]. Further approaches that integrate symbolic and geometric reasoning are presented by Cambon, Alami and Gravot [7], handling geometric preconditions and effects; Dornhege et al. [9]; and, more recently, Plaku and Hager [8], which additionally allow differential motion constraints in a sampling-based motion and action planner. We note that the latter three approaches assume a closed world, where all symbols must be either true or false. On the contrary, our approach represents knowledge in an open-world manner, which allow us to model incomplete knowledge and information-gathering actions. We elaborate on the advantages of this representation in Sec. II-B.

II. Approach

In our work, sensing in robot task planning is seen as a necessary precondition for manipulation and, as such, requires an integrated approach with solutions from distinct fields ranging from motion planning to formal methods. In particular, our KVP framework combines several of these techniques. Symbolic task planning, which includes information-gathering actions and contingencies, is performed by the PKS planner [2], [3], details of which are given below. Motion planning and collision detection rely heavily on the Robotics Library (RL)[1] [15], with several crucial additions to swept volume computation with sets of convex bodies [1]. In order to efficiently generate these sets of convex bodies, Mamou and Ghorbel's approximate convex decomposition algorithm [16] is applied. An overview of KVP's component architecture is shown in Fig. 2 and its implementation is discussed in depth in [1].

A. Planning with Knowledge and Sensing

Symbolic planning in KVP relies on the general-purpose PKS planner, which constructs plans in the presence of incomplete information and sensing actions. PKS works at the *knowledge-level* by reasoning about how the planner's knowledge state, rather than the world state, changes due to action. PKS works with a restricted subset of a first-order language, allowing it to support a rich representation with features such as functions and run-time variables. This approach differs from planners that work with possible worlds models or sets of worlds forming belief states.

PKS is based on a generalization of STRIPS [17]. Unlike STRIPS, which uses a single database to model the world state, PKS's knowledge state is represented by five databases, each of which models a particular type of knowledge. Actions can modify any of these databases, which has the effect of updating the planner's knowledge state. To ensure efficient inference, PKS restricts the type of knowledge (especially disjunctions) that it can represent in each database:

K_f: This database is like a standard STRIPS database except that both positive and negative facts are permitted and the closed-world assumption is not applied. K_f is used to model action effects that change the world. K_f can include any ground literal ℓ, where $\ell \in K_f$ means "the planner knows ℓ." K_f can also contain known function (in)equality mappings.

K_w: This database models the plan-time effects of sensing actions that return binary values. A formula $\phi \in K_w$ means that at plan time, the planner knows whether ϕ or $\neg\phi$ holds, and that at run time this disjunction will be resolved. The use of K_w for robot sensing is described in detail below.

K_v: This database stores information about function values that will become known at execution-time. In particular, K_v can model the plan-time effects of sensing actions that return constants. K_v can contain any unnested function term f, where $f \in K_v$ means the planner "knows the value of f."

PKS also includes databases for modelling a restricted type of disjunctive information (K_x) and "local closed-world" information (LCW), and a mechanism for denoting interval-bounded values which is useful for reasoning about noisy sensors. These features are not used in this paper.

PKS knowledge states can be queried in three different ways. First, simple knowledge assertions can be tested with a query $K(\phi)$ which asks: "is a formula ϕ true?" Second, a query $K_w(\phi)$ asks whether ϕ is known to be true or known to be false (i.e., does the planner "know whether ϕ"). Finally, $K_v(t)$ asks "is the value of function t known?" The negation of the above queries can also be used.

Using this representation, symbolic actions are defined in PKS by describing their (typed) parameters, preconditions, and effects. Preconditions contain a list of queries that must evaluate as true before an action can be applied. Effects are described by a list of `add` and `del` operations, similar to STRIPS. Example PKS actions are shown below in Table I.

B. Contingency Planning for Robot Sensing

One aspect of PKS that is particularly important for robot task planning is its ability to model *sensing actions* that return information about the state of the world. In particular, PKS offers two databases, K_w and K_v, that represent unknown information (binary or function values, respectively) that will become known at run time after the sensing actions

[1]`http://roblib.sf.net/` (accessed Mar. 17, 2013)

TABLE I

FORCE SENSING SCENARIO: EXAMPLE ACTIONS AND GOAL DEFINTION

```
action senseWeight(?o:object)
    preconds:
        ¬$K_w$(isSpillable(?o)) &
        K(isGrasped(?o))
    effects:
        add($K_w$, isSpillable(?o))

action transferUpright(?o:object)
    preconds:
        K(isSpillable(?o)) &
        K(isGrasped(?o)) &
        K(!isRemoved(?o))
    effects:
        add(Kf, isRemoved(?o))

goal: forallK(?o:object)
        (K(isRemoved(?o)) | K(¬isGrasped(?o)))
```

TABLE II

FORCE SENSING SCENARIO: EXAMPLE SOLUTION

```
grasp(can1)
senseWeight(can1)
branch isSpillable(can1)
    transferUpright(can1)
    ungrasp(can1)
    grasp(can2)
    senseWeight(can2)
    branch isSpillable(can2)
        transferUpright(can2)
        ungrasp(can2)
    branch ¬isSpillable(can2)
        transfer(can2)
        ungrasp(can2)
branch ¬isSpillable(can1)
    transfer(can1)
    ...
```

are actually executed in the world. Using these databases, PKS can reason about the possible outcomes of sensing actions during plan construction, by generating plans with *contingencies* (conditional branches).

For instance, in general, PKS builds plans by reasoning about actions in a forward-chaining manner: if the preconditions of an action are satisfied by the planner's knowledge state, then the effects of that action can be applied to produce a new knowledge state. If a formula ϕ is in the K_w database, denoting the fact that ϕ or $\neg\phi$ will become known at run time, then a pair of conditional branches can be added to a plan, with one branch assuming ϕ is true and the other branch assuming $\neg\phi$ is true. (The construction of contingent plans using K_v is similar.) Planning then continues along each branch until the `goal` conditions (a set of queries) are satisfied. A sample plan with branches is shown in Table II, and described in greater detail below.

III. EVALUATION

We now demonstrate and evaluate our approach in two simple scenarios. In the FORCE SENSING scenario (Fig. 1), a compliant robot manipulator has the ability to grasp, lift, and transfer beverage containers which are located on a table. When a container is lifted, the robot can sense its weight and, from this, reason whether the drink must be held upright in order to prevent spilling. The goal is to transfer all containers to a second table, and the robot may hold its gripper upright during these motions, if required. In order to keep this scenario simple, the location of all objects are known and no sensing except force sensing is available.

The second scenario is a demonstration of VISUAL SENSING (Fig. 3) using a bimanual robot whose hands can reach different areas of a table. In this case, the robot can sense if bottles on the table are empty or full using a top-down camera. The goal is to "clean up" all empty bottles and remove them to a certain "dishwasher" location. To do this, the robot must move objects that are only accessible by its left arm to a location that its right arm can reach, a behavior which arises purely from symbolic planning.

In the following sections, we discuss the symbolic domain definitions of both scenarios. We provide an example solution for the first scenario, using conditional branches, and discuss aspects of the plan-based solution for the second scenario.

A. Force Sensing Scenario

Table I shows two PKS actions in the FORCE SENSING scenario, which includes an action `senseWeight` which senses a beverage container `?o`. To perform this action, the robot must first be grasping the object and must not already know whether it is spillable (which acts as an efficiency condition to ensure only new knowledge is gained from this action). When this action is performed, knowledge of whether `?o` is spillable or not is added to PKS's K_w database.

An example manipulation action, `transferUpright`, is also shown in Table I. Using this action, objects that are grasped and not yet "removed" to the second table can be transferred. Besides `transferUpright`, which only handles objects that can be spilled, the complete domain definition also contains a similar action `transfer`, handling all other objects, as well as the necessary `grasp` and `ungrasp` actions that precede and follow the transfer actions.

Table II shown an example contingent plan generated at plan time. The KVP architecture (Fig. 2) executes the actions in this plan and chooses appropriate branches to follow by assessing the results of sensed information. This scenario was physically evaluated on a joint-impedance controlled light-weight 7-DoF robot with a force-controlled parallel gripper. Force was measured by internal torque sensing.

B. Visual Sensing scenario

In contrast to the previous scenario, visual information in this domain (defined in Table III) can be gathered independently from manipulation actions. In this case, the sensing action `senseIfEmpty` has no precondition other than the requirement that the knowledge it gathers must be new. An example manipulation action, `pickUp`, is also shown. Since this scenario contains two robot manipulators, and not all locations can be reached by both hands, the preconditions define an external call `isReachable` to the motion planning component to check reachability for a specific manipulator

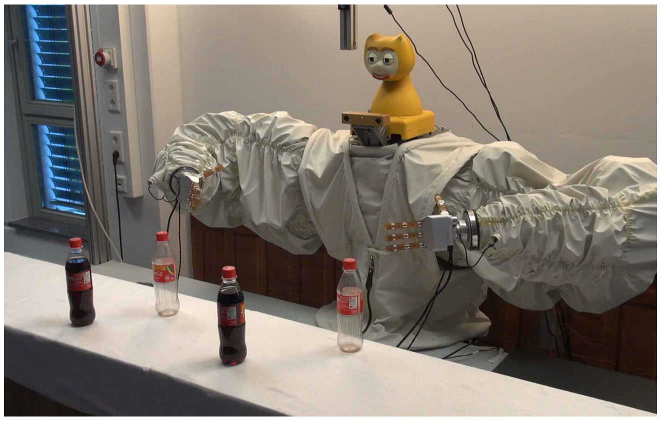

Fig. 3. In the VISUAL SENSING scenario, a camera is used to recognize empty bottles, which a bimanual robot is supposed to remove from the table to a "dishwasher" location on the left side, behind the table [1].

TABLE III

VISUAL SENSING SCENARIO: EXAMPLE ACTIONS AND GOAL

```
action senseIfEmpty(?o:object)
    preconds:
        $\neg K_w$(isEmptyBottle(?o))
    effects:
        add($K_w$, isEmptyBottle(?o))

action pickUp(?r:robot, ?o:object, ?l:location)
    preconds:
        K(?l = getObjectLocation(?o)) &
        K(handEmpty(?r)) &
        K(extern(isReachable(?l, ?r)))
    effects:
        del($K_f$, ?l = getObjectLocation(?o)),
        del($K_f$, handEmpty(?r)),
        add($K_f$, inHand(?o, ?r))

goal: forallK(?o:object)
        (K(getObjectLocation(?o) = dishwasher) |
         K($\neg$isEmptyBottle(?o)))
```

and location. This interaction of the symbolic and motion planners is described in detail in [1]. Evaluation was performed on a two 6-DoF industrial manipulator setup with Meka Robotics H2 humanoid hands, with an RGB camera facing top-down for simple color-filtering object recognition, as described in [18].

It is interesting to observe that this simple bimanual robot scenario already gives rise to interesting behavior: since the right arm cannot directly reach all objects that need to be transferred to the goal location, the left arm must pass those objects to a location reachable by both hands. This behavior has not been pre-programmed, but rather arises purely from symbolic and geometric planning.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduce an approach to task planning with sensing actions, incomplete information, and multiple manipulators and sensors, using the PKS planner and the KVP framework. We illustrate the effectiveness of this approach on two simple scenarios that cover force sensing and visual sensing, with real execution on physical robot setups.

As future work, we plan to generalize our symbolic approach to task space constraints for object manipulation, and explore more efficient heuristic search strategies at the symbolic planning level, including building plans with loops.

V. ACKNOWLEDGMENTS

This research was supported in part by the European Union's Seventh Framework Programme through the JAMES project (Joint Action for Multimodal Embodied Social Systems) under grant agreement no. 270435.[2]

REFERENCES

[1] A. Gaschler, R. Petrick, M. Rickert, M. Giuliani, and A. Knoll, "KVP: A knowledge of volumes approach to robot task planning," 2013, submitted.

[2] R. Petrick and F. Bacchus, "A Knowledge-Based Approach to Planning with Incomplete Information and Sensing," in *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 2002, pp. 212–221.

[3] ——, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proc. of the Int. Conference on Automated Planning and Scheduling (ICAPS)*, 2004, pp. 2–11.

[4] N. Nilsson, "Shakey The Robot," AI Center, SRI International, Tech. Rep. 323, Apr. 1984.

[5] T. Lozano-Pérez, J. Jones, E. Mazer, and P. O'Donnell, "Task-level planning of pick-and-place robot motions," *Computer*, vol. 22, no. 3, pp. 21–29, 1989.

[6] L. Kaelbling and T. Lozano-Pérez, "Unifying Perception, Estimation and Action for Mobile Manipulation via Belief Space Planning," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2952–2959.

[7] S. Cambon, R. Alami, and F. Gravot, "A Hybrid Approach to Intricate Motion, Manipulation and Task Planning," *Int. Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[8] E. Plaku and G. Hager, "Sampling-based motion planning with symbolic, geometric, and differential constraints," in *IEEE Int. Conference on Robotics and Automation*, 2010, pp. 5002–5008.

[9] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating symbolic and geometric planning for mobile manipulation," in *IEEE Intl. Workshop on Safety, Security & Rescue Robotics*, 2009, pp. 1–6.

[10] C. Cheng, M. Geisinger, H. Ruess, C. Buckl, and A. Knoll, "Game Solving for Industrial Automation and Control," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 4367–4372.

[11] S. Chinchali, S. Livingston, U. Topcu, J. Burdick, and R. Murray, "Towards Formal Synthesis of Reactive Controllers for Dexterous Robotic Manipulation," in *IEEE Int. Conference on Robotics and Automation*, 2012, pp. 5183–5189.

[12] F. Zacharias, C. Borst, and G. Hirzinger, "Bridging the gap between task planning and path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4490–4495.

[13] L. Kaelbling and T. Lozano-Pérez, "Hierarchical Task and Motion Planning in the Now," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1470–1477.

[14] R. Petrick, D. Kraft, N. Krüger, and M. Steedman, "Combining Cognitive Vision, Knowledge-Level Planning with Sensing, and Execution Monitoring for Effective Robot Control," in *Workshop on Planning and Plan Execution for Real-World Systems*, 2009, pp. 58–65.

[15] M. Rickert, "Efficient Motion Planning for Intuitive Task Execution in Modular Manipulation Systems," Dissertation, Technische Universität München, 2011.

[16] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3D mesh approximate convex decomposition," in *IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3501–3504.

[17] R. Fikes and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[18] M. E. Foster, A. Gaschler, M. Giuliani, A. Isard, M. Pateraki, and R. Petrick, "Two People Walk Into a Bar: Dynamic Multi-Party Social Interaction with a Robot Agent," in *Proceedings of the ACM International Conference on Multimodal Interaction (ICMI)*, 2012.

[2]http://www.james-project.eu/ (accessed Mar. 17, 2013)