2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

KVP: A Knowledge of Volumes Approach to Robot Task Planning

Andre Gaschler, Ronald P. A. Petrick, Manuel Giuliani, Markus Rickert and Alois Knoll

***Abstract*— Robot task planning is an inherently challenging problem, as it covers both continuous-space geometric reasoning about robot motion and perception, as well as purely symbolic knowledge about actions and objects. This paper presents a novel "knowledge of volumes" framework for solving generic robot tasks in partially known environments. In particular, this approach (abbreviated, KVP) combines the power of symbolic, knowledge-level AI planning with the efficient computation of volumes, which serve as an intermediate representation for both robot action and perception. While we demonstrate the effectiveness of our framework in a bimanual robot bartender scenario, our approach is also more generally applicable to tasks in automation and mobile manipulation, involving arbitrary numbers of manipulators.**

I. Introduction

In recent years, symbolic task planners have made substantial progress in their ability to reason about incomplete knowledge and perceptual information [1], [2], [3], [4], laying the foundation for modelling realistic robot environments. One promising technique is the knowledge-based planning approach [5], [6], which enables a robot to reason about unknown state information and the effects of perceptual actions, thereby providing the tools needed for robust planning when a robot may only have incomplete information about its environment. However, integrating the geometric properties of the robot and the environment into a purely symbolic task planner poses significant challenges: robot systems must reason about joint angles, spatial coordinates and physical bodies in continuous spaces; while high-level task planners typically rely on discrete, symbolic representations of features, values, and actions described in logical languages [2]. As a result, very few approaches have successfully bridged the gap between geometric and symbolic representations, allowing logical task planners to work effectively with geometric constraints [7], [8].

In this paper, we describe a *knowledge of volumes approach to robot task planning* (abbreviated, *KVP*), which treats volumes as an intermediary representation between continuous-valued robot motions and discrete symbolic actions. An off-the-shelf, general purpose, symbolic AI planner called PKS (Planning with Knowledge and Sensing; [6], [9]) is employed as a backend reasoning engine for efficiently computing knowledge-level task plans, utilising the volume-based representation in the underlying description of the robot's operational domain. By combining these two techniques, the robot can reason with incomplete information about objects, motions, and actuation actions, and perform information-gathering sensing actions when necessary. This gives rise to a novel approach to robot task planning which has certain inherent advantages:

1) 3D geometric volumes are a natural intermediate representation, making it suitable for bridging the gap between motion planning and high-level task planning.
2) Continuous geometry is preferable over indiscriminate discretisation [8]: novel geometric simplification techniques [10] allow computationally efficient collision detection of volumes [11], enabling continuous geometry planning to be applied in real systems.
3) Knowledge-based planning naturally allows reasoning about acting and sensing in a structured, partially known environment, which is reflected by concise and clear domain descriptions, keeping the overall approach generic for a broad set of robotics applications.

As a result, this work makes a number of strong contributions to the problem of combining continuous-space geometric reasoning with high-level symbolic planning. Our approach is among the first to use 3D geometric volumes as the underlying representation for symbolic planning and motion planning, and the first to combine this idea with an off-the-shelf, general-purpose AI planner supporting deterministic planning with incomplete information and sensing. In contrast to more specialised approaches, which limit the scope of application, our use of a general-purpose planner provides us with a tool that can be applied in contexts beyond that of the demonstrated bimanual robot, including tasks in automation and mobile manipulation, involving arbitrary numbers of manipulators. Moreover, by building on standard planning representations, we can take advantage of new planning engines that become available from the planning

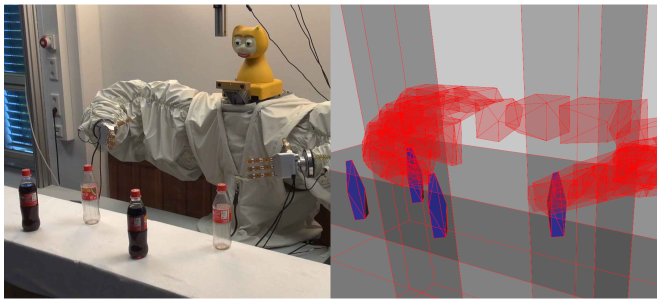


Fig. 1. Typical robot tasks can be modelled as symbolic actions and geometric volumes, representing both swept volumes of robot motions and object boundaries.

A. Gaschler, M. Giuliani and M. Rickert are 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.

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

community, which may be better optimised for our task. Finally, our work is grounded in real-world application, and all of our experiments are tested on actual robot platforms, rather than solely in simulated environments.

The rest of this paper is structured as follows. In Section II, we compare our work to related approaches, and then describe our framework in greater detail in Section III. In Section IV, we evaluate our approach in a real robot domain, namely a two-handed robot bartender scenario. Finally, in Section V, we discuss future work and conclude.

II. Related Work

Some of the earliest work on robot task planning dates back to systems like Shakey [12] and Handey [13]. Since those early approaches, the field has made significant developments, with interest in the problem of real-world task planning gaining momentum in recent years. In particular, the problem has been explored from several diverse areas of research, including probabilistic models from AI [14], closed-world symbolic planning [7], [15], [16], formal synthesis [17], [18], [19], and manipulation planning [20].

A very recent contribution, most closely related to our approach, is the belief space planner by Kaelbling and Lozano-Pérez [14], which operates on a belief space of probability distributions over states, providing robustness against uncertainty and unexpected change. Perception (and estimation) arises as a necessary precondition for manipulation, rather than being hard-coded as a task itself. Using a simulation of a PR2 mobile manipulator, Kaelbling and Lozano-Pérez demonstrate the effectiveness of their approach through a series of experiments in initially unknown environments. In contrast to their notion of belief, our knowledge-based planning approach relies on discrete logic and abstracted state descriptions, which we believe is a viable alternative in structured environments with certain, but incomplete, information about the world. Moreover, we use an existing planner which has proven successful in previous robot deployments [2], rather than designing a new planner for the task. In terms of volumes, Kaelbling and Lozano-Pérez work with octrees, while we use sets of convex shapes, allowing efficient collision detection in the deterministic case.

More traditionally, high-level task planning is often seen as a computational layer on top of motion planning, and several approaches try to integrate both layers. Our work is in part inspired by Kaelbling and Lozano-Pérez' earlier work on hierarchical task and motion planning in [8], borrowing the continuous geometry of swept volumes. But, whereas the geometric preconditions may be similar, their aggressively hierarchical planning strategy differs from our knowledge-based approach which make use of abstract (but non-hierarchical) structures. Cambon, Alami and Gravot [7], [21] also propose a task planning algorithm that can internally handle geometric preconditions and effects. Their approach systematically integrates symbolic PDDL- (Planning Domain Definition Language) [22] based states and actions together with geometric motion planning. Dornhege et al. [16] also present a mobile manipulation planner, which is similar to the way the planner we use invokes and evaluates geometric functions. More recently, Plaku and Hager [15] develop a similar, sampling-based motion and action planning approach, additionally allowing differential motion constraints.

It is important to note that many of the above approaches assume a closed world, where all symbols are either true or false, which is a significant limitation for many robotic scenarios. In contrast to this closed world assumption, our task planning approach is based on an open-world knowledge representation which naturally models incomplete information and sensing actions—advantages which we elaborate on in Section III-B. It should further be noted that in the related works described above, either the evaluation of the combined task and motion planner is confined to simulation [8], [7], [16], [15], or a general domain-independent planning scheme is not the focus of the work [20] when a real robot system is used to evaluate the approach.

III. Approach

Integrated robot task planning requires a multidisciplinary approach, adapting solutions from different fields ranging from motion planning to formal methods. KVP combines several of these techniques, which we outline below. We begin by discussing the geometric models we use (Section III-A), followed by a description of the knowledge-level PKS planner (Section III-B), and the planning domain we have defined (Section III-C). Finally, we describe the overall system architecture of our framework (Section III-D). Our evaluation scenario is a simple robot bartender that clears away empty bottles from a table (see Figure 1), requiring both sensing and manipulation actions.

A. Convex Decomposition of Volumes

We use volumes as an intermediary representation of a geometric shape, modelled in a high-level symbolic form. In KVP, volumes are used to represent the physical boundaries of both static objects and dynamic motions—as well as the view cone of a sensor, representing an area of perception. It is therefore important to process these volumes using a computationally-efficient data type, for which we use sets of convex bodies. A convex body is defined by a set of 3D points, and the body is the set of all convex combinations of these points. To ease computations, the set of outer triangles (triples of these point indices) is usually saved within the same data object. Even though a single convex body may be too conservative an approximation of the boundaries of most volumes in our domain, a small set of convex bodies usually suffices to accurately approximate volumes as complex as the swept volumes of a robot, as shown in Figure 2. Using sets of convex bodies also leads to a computationally-efficient collision detection process, which is a significant advantage of this approach in real-world applications.

Although the above approach provides a computationally-efficient method that can be used online at run time, the offline task of decomposing arbitrary geometric models into sets of convex shapes remains a challenging problem. For our KVP framework, we chose an implementation of the

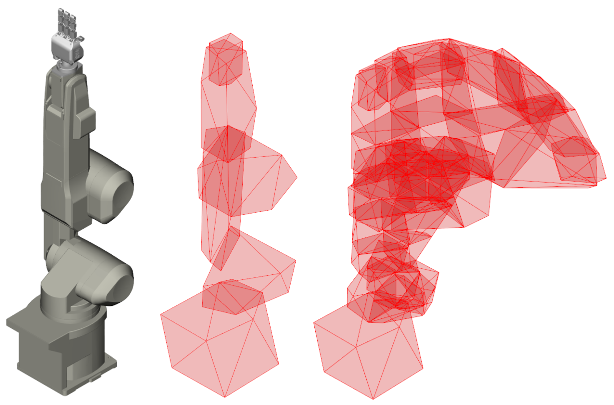


Fig. 2. Convex decomposition allows an efficient approximation of swept volumes. A typical robot mesh has 10^6 vertices and is non-convex (left). Convex decomposition [10] simplifies this to 6 convex bodies with 10 vertices each (centre), allowing a typical swept volume description with only 40 convex bodies, totalling 400 vertices (right) [23].

algorithm in [10] by Mamou and Ghorbel. In particular, their approach performs a hierarchical segmentation on the dual graph of triangles, which is lead by a cost function based on concavity and an aspect ratio measure. As a concavity measure for a given 3D mesh, they choose the maximum distance of all mesh surface points in surface normal direction to their projection onto the convex hull of that mesh. For the aspect ratio, they define the squared perimeter divided by the area of a given 3D mesh, adjusted by a constant factor to yield one in the case of a disk. In the iterative surface simplification, both cost function components are weighted such that aspect ratio dominates the early stage of the algorithm, which quickly compacts the surface, followed by the concavity measure determining the final simplifications. At each iteration, a half-edge collapse decimation is performed on the dual graph.

Mamou and Ghorbel show that their approach is superior to existing solutions, both in terms of the approximation quality and number of convex shapes, as well as speed. As indicated in Figure 2, their algorithm produces a concise set of convex bodies, which are ideal for efficient collision detection. A typical six-axes robot manipulator can be approximated by 6 to 10 convex shapes, totalling no more than 100 points. A swept volume of a typical motion for such a robot will simplify to no more than 20–100 convex bodies, and can be efficiently computed by merging subsequent samples of the robot model along the motion path [23]. In practice, sets of convex bodies provide a sufficiently efficient geometric representation so that collision detection often accounts for only a small fraction of total computation time.

B. Planning with Knowledge and Sensing (PKS)

High-level planning in KVP is provided by the off-the-shelf PKS (Planning with Knowledge and Sensing) planner [6], [9], which is able to construct plans in the presence of incomplete information and sensing actions.[1] PKS works at the knowledge-level [5] by reasoning about how the planner's knowledge, rather than the world state, changes due to action. PKS is a symbolic planner that is built on a subset of a first-order logical language with restricted inference, allowing it to efficiently support a wide range of optimised features that result from limiting its representation language. This approach differs from planners based on possible worlds or belief states, which often trade tractability for more comprehensive representations and reasoning capabilities.

[1]PKS is available from `http://homepages.inf.ed.ac.uk/rpetrick/software/pks/`.

PKS is based on a generalisation of STRIPS [24] and uses a database mechanism as its underlying state representation. In particular, knowledge states in PKS are represented by a set of five databases, each of which models a particular type of knowledge. Actions can modify any of the 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 this work, we mainly focus on three of PKS's databases:

$\mathbf{K_f}$: This database is like a STRIPS database that stores the values of regular fluents that the planner knows. K_f is primarily used for modelling the effects of actions that change the world. Unlike standard STRIPS, it works with an open world model that can explicitly represent both positive and negative facts. In particular, K_f can include any ground literal ℓ, where $\ell \in K_f$ means "the planner knows ℓ."

$\mathbf{K_w}$: This database stores information about the effects of sensing actions that return one of two possible outcomes, providing support for information-gathering actions that observe the world but do not necessarily change it. A formula $\phi \in K_w$ means that at plan time the planner either "knows ϕ or knows $\neg\phi$." However, this disjunction will not be resolved until run time when the action is actually executed.

$\mathbf{K_v}$: This database stores information about function values that will become known at execution time. In particular, K_v can model the effects of sensing actions that return one of many possible values. K_v can contain any 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) [25], which are not used in this paper.

Reasoning in PKS is done through a limited set of primitive queries that ask simple questions about the planner's knowledge state: (i) is a fact ϕ known to be true (i.e., $K(\phi)$) or known to be false (i.e., $K(\neg\phi)$)? (ii) does the planner know whether a property ϕ is true or not (i.e., $K_w(\phi)$)? (iii) is the value of a function t known (i.e., $K_v(t)$)? (iv) or the negation of the above queries. An efficient inference algorithm evaluates primitive queries by checking the contents of the databases and the relationship between the databases. Details of this procedure can be found in the original PKS papers by Petrick and Bacchus [6], [9].

An action in PKS is described by its parameters, preconditions, and effects. An action's parameters are a list of typed variables which may be used throughout the action definition. When evaluated, the planner replaces these parameters with objects of the appropriate type, chosen by the planner, which are bound to each occurrence of the parameter. An action's preconditions are a list of primitive queries which must evaluate as true before an action can be applied. Action effects are described by a collection of STRIPS-style "add" and "delete" operations that allow information to be inserted and removed from individual databases. For instance, $add(K_f, \phi)$ adds ϕ to K_f, and $del(K_w, \phi)$ removes ϕ from K_w. Two PKS actions from our evaluation domain are shown in Table I, which we will discuss in greater detail in Section III-C.

An important class of actions that PKS can model are sensing actions that return information about the state of the world. In PKS, sensing actions are specified by effects that update the K_w or K_v databases (i.e., databases that track properties with multiple potential outcomes). Given such information, the planner can build contingencies into a plan by introducing a conditional plan branch for each possible outcome of the sensing action. (E.g., the `senseIfEmpty(?o)` action in Table I is an example of a sensing action in our evaluation domain that provides information about the state of `isEmptyBottle(?o)`, which can be true or false.)

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 action's effects are applied to produce a new knowledge state. Planning then continues from this new state. The choice of which action to apply next in a given state is achieved by using either a depth-first search or breadth-first search strategy, optimised by a set of heuristics. (Depth-first search typically produces plans more quickly than breadth-first search, however, breadth-first search generally results in plans with fewer steps.) PKS can also build plans with contingencies by considering the potential outcomes of a sensing action (i.e., its K_w and K_v knowledge). For instance, if ϕ is in K_w then PKS can introduce two branches into a plan: along one branch ϕ is assumed to be true, while along the other branch $\neg\phi$ is assumed to be true. Planning continues along each branch until the goal conditions (a set of primitive queries) are satisfied in each case.

One important feature of PKS, central to the KVP framework, is its ability to integrate externally-defined procedures (e.g., from support libraries) with its internal reasoning mechanisms [26]. A special keyword, `extern`, supports this facility, where an expression of the form `extern`$(proc(\vec{x}))$ means that the parameters $\vec{x}$ should be passed to an external procedure $proc$ for execution. $\vec{x}$ can contain symbols defined in the planning domain, providing a link between the planner and the externally-defined procedure. The return value of an `extern` call is passed back to PKS, which can then perform additional tests on this value, or include it in its knowledge base. (E.g., the `pickUp` action in Table I uses an `extern` call to invoke a path planner.) An `extern` call can also be directed to cache its return value for efficiency. PKS's ability to invoke external procedures is key to our KVP approach, enabling us to augment PKS's reasoning capabilities through motion planning, collision detection, and other special purpose robotics libraries.

TABLE I
EXAMPLE PKS ACTIONS AND GOALS FOR THE BARTENDER SCENARIO

```
action senseIfEmpty(?o:object)
   preconds:
      ¬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(¬isEmptyBottle(?o)))
```

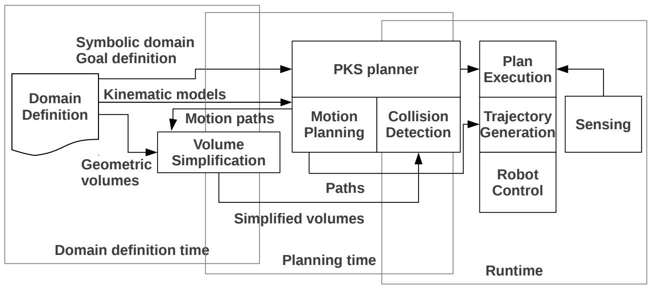


Fig. 3. Overview of the implemented KVP software architecture.

C. Planning Domain Definition

The experimental application domain for this work is a robot bartender scenario, where the robot can manipulate objects (e.g., bottles) in the world. In order to use PKS in this scenario we must first supply a symbolic domain description, characterising its properties and actions, and the robot's goal.

Properties in PKS are similar to logical predicates and functions, and may have typed arguments. For our experimental domain we define three types: `robot`, `object`, and `location`. Using these types, we define predicates that characterise the domain's state space: `isEmptyBottle(?o)` (object `?o` is an empty bottle), `handEmpty(?r)` (robot hand `?r` is empty), `inHand(?o,?r)` (object `?o` is in robot hand `?r`), and `isReachable(?l,?r)` (location `?l` is reachable by robot hand `?r`). We also include a function, `getObjectLocation(?o)=?l` (object `?o` is in location `?l`).

Using these symbols, we can formulate PKS actions, as described above in Section III-B. Table I shows an example of two PKS actions in the bartender scenario. Here, `senseIfEmpty(?o)` is a sensing action that determines whether or not a bottle `?o` is empty. This information-gathering action is modelled by an effect that adds the `isEmptyBottle(?o)` predicate to the K_w database, provided this information isn't already known to the planner.

The inclusion of senseIfEmpty(?o) allows the planner to build contingent branches in its plan, where each branch considers one of the possible outcomes of isEmptyBottle(?o) (i.e., one contingency if isEmptyBottle(?o) is true, and another if isEmptyBottle(?o) is false). However, at run time, when the true value of isEmptyBottle(?o) is known, the appropriate branch of the plan will be executed on the robot. A more detailed description of sensing actions and branched plans in this domain is given in [27].

Table I also includes the action pickUp(?r,?o,?l), a physical action that uses robot hand ?r to pick up object ?o from location ?l. This action is modelled by a set of preconditions that verify the planner knows the location of ?o is ?l, that hand ?r is empty, and that location ?l is reachable with ?r. When the action is applied, the planner comes to know that ?o is no longer at ?l, hand ?r is no longer empty, and that the robot is holding ?o in hand ?r.

For a typical robot action, motion planning often needs to be performed. To specify such behaviour as part of the pickUp action, we include an external call to the procedure isReachable(?l,?r) (indicated by the extern directive), which directs PKS to invoke the path planner for a particular location ?l and robot hand ?r, and report if a path could be generated. This allows us to define actions to pick and place objects. In contrast to the sensing action senseIfEmpty, whose outcome is evaluated at run time, the motion planning procedure isReachable is evaluated at planning time.

The complete PKS domain description also includes an action putDown(?r,?o,?l), which allows the robot to place an object ?o at a location ?l using hand ?r. This action is defined in a similar manner to the pickUp action in Table I.

The final component of our domain definition is a specification of the robot's goal. In our bartender scenario, the robot is given the task of moving all empty bottles to a location named dishwasher, specified by the goal expression in Table I. The goal is achieved if each object is either in the dishwasher, or isn't an empty bottle. With the slight addition of swept volume collision checking to avoid collisions between robot hands (which we omit for clarity of presentation), we obtain the domain definition used in the evaluation in Section IV below. We note that even though we use the bartender scenario as an example of our approach, this domain description may easily be generalised to other robot tasks as it involves common object transfers and collision-avoiding robot motions.

D. System Architecture

The components described above—volume simplification, the PKS planner, and the planning domain description—account for only part of the KVP platform. To test the effectiveness of our approach, we have implemented and evaluated our framework both in simulation and on a real robot system. The complete system architecture is shown in Figure 3. Although this architecture contains a number of support modules which aid the components described above, the only computationally expensive task in this framework is volume simplification, which can mostly operate offline: the static scene, all object types, and the robot limbs need to be simplified only once. Swept volumes need to be computed as part of the planning process, and can be simplified by using the previously calculated robot limb volumes.

During planning, PKS can generate motion plans and check for collisions by directly calling functions from the motion planning and collision detection components. Both of these components rely heavily on the Robotics Library (RL)[2] by Rickert [28]. RL includes several efficient motion planning algorithms and a manually optimised algebraic solution of the inverse kinematics for a broad range of six degrees-of-freedom robots, including the industrial manipulator used in the evaluation. For path planning, we apply a simple interpolation in configuration space between locations in the defined domain. We also include a simple grasp planner that can evaluate a set of grasping poses of the end effector.

In the evaluated bartender scenario (see below), the grasping poses are generated from one defined end effector pose sampled around a bottle's axis of rotation in 30 degree steps. During path planning, paths are only checked for collisions with the static environment. All other collision checks are listed in the action preconditions and dispatched by PKS, calling the collision checker when necessary. The actual collision detection is based on the Bullet Physics Library,[3] which is geared towards efficiency and can handle large numbers of convex bodies using fast broad-phase detection.

Action execution at run time is mediated by a simple plan execution component, which processes the plans generated by PKS, one action at a time. For robot actions, plan execution calls a trajectory generator, which performs a quintic interpolation of the robot paths in configuration space. Once generated, these trajectories are then executed on the physical robot. For sensing actions that give rise to contingent (branching) plans, the outcome of the sensing is used to determine which contingency (branch) of the plan should be followed [27]. In our evaluation scenario, the only run-time sensing action involves a simple colour-based vision component (see Figure 4) that reports if a bottle is full or empty. We note that the spatial locations of the bottles are also visually detected by this sensing component. However, since this information is also needed for motion planning, it may already be available at planning time. The plan execution component also acts as a recovery mechanism that allows plans to be rebuilt in response to unexpected changes in the environment. For example, in our bartender domain, bottles that are newly detected at run time could trigger a replanning stage, which would lead to new (but few) geometric volume computations. While we do not use replanning in our evaluated system, in general, this mechanism provides a useful tool that can improve the overall robustness of the system in many domains.

IV. Evaluation

We performed an evaluation of our KVP implementation both in simulation and on a bimanual robot system, whose

[2]http://roblib.sf.net/
[3]http://bulletphysics.org/

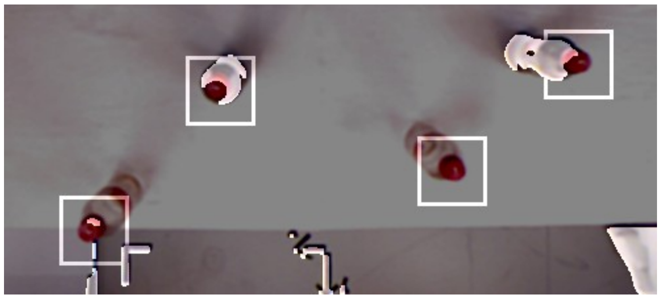


Fig. 4. Run-time sensing in the evaluated scenario includes simple colour-based segmentation for empty and full bottles. [27]

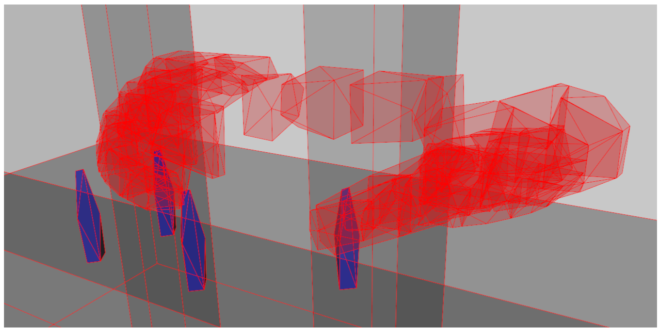


Fig. 5. World model with two swept volumes of grasping motions (red), movable objects (blue), and static obstacles (grey).

components are described in [29]. The robot setup consists of two 6-DoF industrial manipulators with Meka Robotics H2 humanoid hands. Visual input is given by a depth sensing device mounted on top of the setup. The goal in this scenario is to detect bottles located on the bar and remove all empty bottles to a special `dishwasher` location at the right of the robot. The three PKS actions available in this domain (`pickUp`, `putDown`, and `senseIfEmpty`), are defined as discussed above. Geometric data is confined to 3D models of the robot, a bottle, and the bar environment, as indicated in Figure 5. Finally, kinematic data of the two robot arms and a simple grasping scheme is also included.

The planning and robot control software ran on a 2.8 GHz desktop computer. The computer vision component ran on a separate machine and the robot hands were controlled by a separate real-time PC. The robot could execute the task without human intervention. (For details, see Figure 6 and the associated video.) Even in this simple scenario, KVP must find a non-trivial solution. In order to move bottles from the right side of the bar to the goal location, one arm must first move them to a location in reach of the second arm. Using the symbolic definition of volumes, all possible collisions are avoided, including less obvious cases.

Table II shows the running times for various aspects of KVP, under alternative search methods used by the PKS planner for plan generation. While the results indicate depth-first search is more efficient than breadth-first search for scenarios more complicated than our evaluation domain, breadth-first search may yield significantly shorter plans. Furthermore, our results illustrate that the total planning time is adequately short for the evaluated scenario, and does not significantly impact on the overall execution of the system. For depth-first search, planning is an order of magnitude faster than both the offline simplification of static volumes and the actual execution of robot actions, making it an efficient technique for solving problems in this scenario.

TABLE II
EFFICIENCY OF KVP FOR THE BARTENDER SCENARIO

	Depth-first search	Breadth-first search
Total domain definition time, of which:	27.399 s	27.389 s
Volume simplification	27.221 s	27.221 s
Total planning time, of which:	2.908 s	36.673 s
Motion planning of inverse kinematics	0.036 s	0.572 s
Robot volume computations	0.578 s	0.568 s
Swept volume simplification	2.251 s	34.949 s
Collision checking	0.001 s	0.002 s
Function calls		
Motion planner path generation	8	16
Collision checking	20	153
Run-time execution time	68 s	68 s

V. Conclusion and Future Work

In this paper, we describe an approach to robot task planning that combines reasoning about complex geometric volumes with general-purpose knowledge-level planning techniques. We demonstrate the effectiveness of our KVP framework on a realistic two-robot setup that includes run-time perception and physical robot actions in a bartender scenario. Overall, we believe that volume-based task planning is applicable to a broad range of robot tasks, and may prove effective in structured and partially known environments, including automation, robot-aided manufacturing, and mobile manipulation, involving arbitrary numbers of manipulators.

We also view the use of a general-purpose planner that has not been explicitly optimised for planning in robotics domains as an advantage of our approach. First, our framework profits from future updates and improvements to the actively-developed PKS planner. Second, by treating planning as a black box we keep our framework sufficiently modular, allowing us to consider other symbolic planners from the planning community which can be substituted for PKS and tested in the KVP framework with minimal effort.

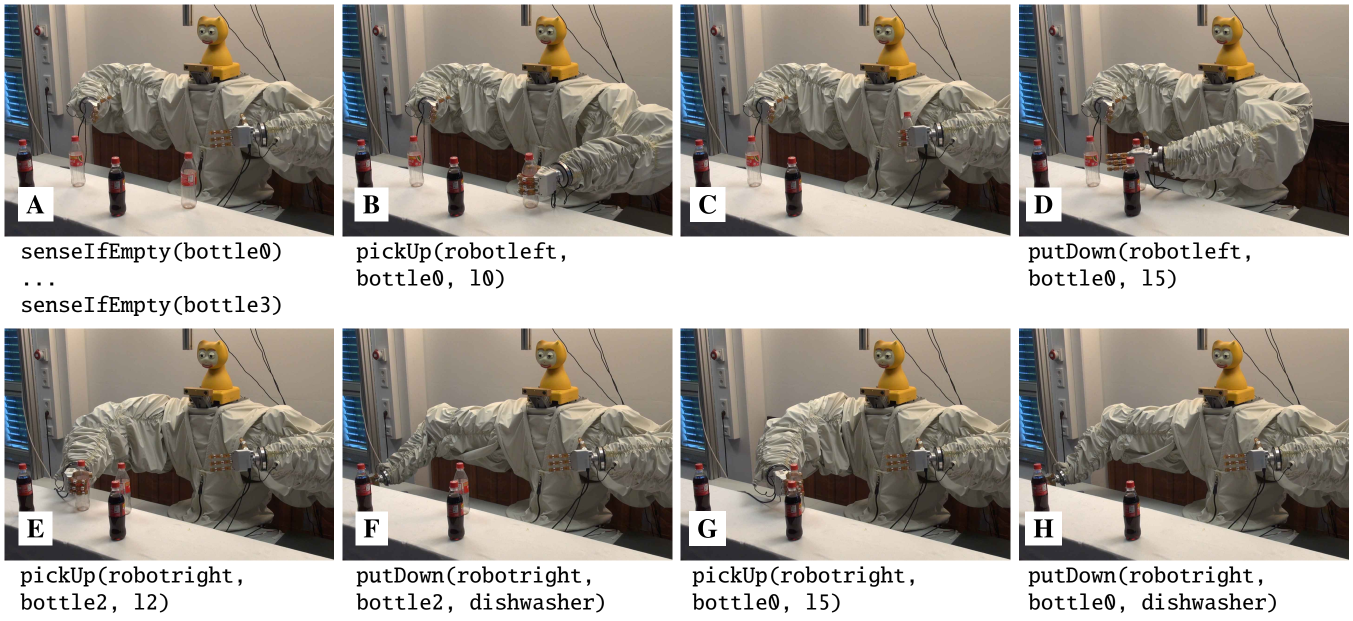
As future work, we plan to explore other symbolic planners, as well as extensions to PKS to improve its performance in more complex robotics domains. We will also investigate the notion of volumes for perception further, which neatly fits into the KVP framework, and may enable us to identify initially unknown or dynamic objects and obstacles.

References

[1] J. Hoffmann and R. Brafman, "Contingent Planning via Heuristic Forward Search with Implicit Belief States," in *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2005, pp. 71–80.

[2] R. Petrick, D. Kraft, K. Mourão, C. Geib, N. Pugeault, N. Krüger, and M. Steedman, "Representation and integration: Combining robot control, high-level planning, and action learning," in *Proceedings of the Int. Cognitive Robotics Workshop (CogRob)*, 2008, pp. 32–41.

[3] A. Albore, H. Palacios, and H. Geffner, "A translation-based approach to contingent planning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 1623–1628.



```
senseIfEmpty(bottle0)
...
senseIfEmpty(bottle3)

pickUp(robotleft,
bottle0, l0)

putDown(robotleft,
bottle0, l5)

pickUp(robotright,
bottle2, l2)

putDown(robotright,
bottle2, dishwasher)

pickUp(robotright,
bottle0, l5)

putDown(robotright,
bottle0, dishwasher)
```

Fig. 6. Action sequence of a solution in the evaluation scenario: In this example, the left arm is the only one to reach the bottle (image **B**). However, only the right arm can reach the desired `dishwasher` location. Therefore, the robot's left arm moves the first bottle to a location where the other arm can reach it (image **D**), and the right arm passes it on to its final location (images **G** and **H**). We note that this behaviour has not been pre-programmed but, rather, arises purely from symbolic planning.
A video on this scenario is available in the file attachment of this paper and on the webpage `http://youtu.be/yMmZkhHr8ss`.

[4] M. Göbelbecker, C. Gretton, and R. Dearden, "A Switching Planner for Combined Task and Observation Planning," in *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, 2011, pp. 964–970.

[5] A. Newell, "The Knowledge Level," *Artificial Intelligence*, vol. 18, no. 1, pp. 87–127, 1982.

[6] 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.

[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] 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.

[9] R. Petrick and F. Bacchus, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2004, pp. 2–11.

[10] 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.

[11] S. Redon, M. Lin, D. Manocha, and Y. Kim, "Fast continuous collision detection for articulated models," *Journal of Computing and Information Science in Engineering*, vol. 5, p. 126, 2005.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] H. Kress-Gazit and G. Pappas, "Automatically synthesizing a planning and control subsystem for the DARPA Urban Challenge," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, 2008, pp. 766–771.

[18] 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.

[19] 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.

[20] 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.

[21] F. Gravot, S. Cambon, and R. Alami, "aSyMov: a planner that deals with intricate symbolic and geometric problems," *Robotics Research*, pp. 100–110, 2005.

[22] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL – The Planning Domain Definition Language (Ver. 1.2)," Yale Center for Computational Vision and Control, Technical Report CVC TR-98-003/DCS TR-1165, 1998.

[23] A. Gaschler, R. P. A. Petrick, T. Kröger, O. Khatib, and A. Knoll, "Robot Task and Motion Planning with Sets of Convex Polyhedra," in *Robotics: Science and Systems (RSS) Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[24] 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.

[25] O. Etzioni, K. Golden, and D. Weld, "Tractable Closed World Reasoning with Updates," in *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR)*, 1994, pp. 178–189.

[26] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic Attachments for Domain-Independent Planning Systems," in *Proc. of the Int. Conference on Automated Planning and Scheduling (ICAPS)*, 2009, pp. 114–121.

[27] A. Gaschler, R. P. A. Petrick, T. Kröger, A. Knoll, and O. Khatib, "Robot Task Planning with Contingencies for Run-time Sensing," in *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Combining Task and Motion Planning*, 2013.

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

[29] 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.