Robot Task and Motion Planning with Sets of Convex Polyhedra

Andre Gaschler*, Ronald P. A. Petrick†, Torsten Kröger‡, Oussama Khatib‡, Alois Knoll*

*fortiss An-Institut der TU München, Munich, Germany, Email: gaschler@fortiss.org
†School of Informatics, University of Edinburgh, Edinburgh, United Kingdom, Email: rpetrick@inf.ed.ac.uk
‡Artificial Intelligence Laboratory, Stanford University, Stanford, USA

***Abstract*—Geometric volumes can be used as an intermediate representation for bridging the gap between task planning, with its symbolic preconditions and effects, and motion planning, with its continuous-space geometry. In this work, we use sets of convex polyhedra to represent the boundaries of objects, robot manipulators, and swept volumes of robot motions. We apply efficient algorithms for convex decomposition, conservative swept volume approximation and collision detection, and integrate these methods into our existing "knowledge of volumes" approach to robot task planning called KVP. We demonstrate and evaluate our approach in several task planning scenarios, including a bimanual robot platform.**

I. Introduction

The problem of combining task planning and motion planning in a robot system presents significant representational difficulties that must be overcome: high-level task planners typically rely on symbolic representations of objects and actions, while motion planning systems need to reason about physical bodies and robot motion in a continuous space. Integrating geometric and symbolic reasoning in a common framework is therefore a challenging and important task.

In this paper, we describe an approach that represents the boundaries of objects, robots, and swept volumes of robot motions as geometric volumes, which serves as an intermediate representation between continuous and discrete reasoning. In particular, this work focuses on algorithms for convex decomposition, conservative swept volume generation, and collision detection, implemented in an existing "knowledge of volumes" approach to robot task planning called KVP [9, 10].

In addition to using volumes as a representation of robot motions and objects, KVP is further characterized by symbolic planning at the knowledge-level, with direct function calls to the robotics component for geometric and kinematic queries. As a symbolic AI planning system, we use the general purpose PKS planner (Planning with Knowledge and Sensing [18, 19]), which allows contingent planning with incomplete information and sensing actions. In particular, this approach enables us to use planned sensing actions as preconditions for manipulation, rather than being hard-coded as tasks themselves [10].

The remainder of the paper is organised as follows. In Section II, we situate our work with respect to related approaches. In Sections III-A and III-B, we analyse the efficiency of our approach, show the quadratic convergence of our swept volume approximation in joint space, and describe a complexity class for collision checking. Details of the symbolic planner are given in Section III-C. Finally, we evaluate the efficiency of the KVP approach in Section IV, in three example domains.

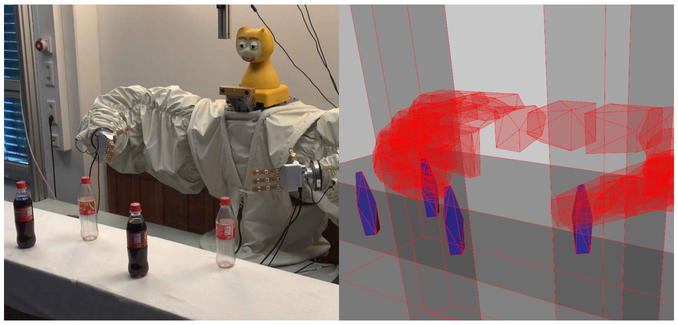
Fig. 1. Robot tasks are modelled by symbolic actions and geometric volumes, representing both swept volumes of robot motions (red), and object boundaries as sets of convex polyhedra (blue for movable objects, grey for static obstacles) [9].

II. Related Work

Although the problem of robot task planning has been investigated since the days of early robotic systems like Shakey [17], more recently the field has gained substantial attention, both from the planning and robotics communities. Modern approaches to robot task planning include a diverse range of techniques from probabilistic models in artificial intelligence [14], closed-world symbolic planning [2, 21], formal synthesis [15, 3], and multimodal motion planning [12].

Our KVP approach is in part inspired by Kaelbling and Lozano-Pérez's work on hierarchical task and motion planning [13], borrowing the continuous geometry of swept volumes. However, while the geometric preconditions may be similar, their underlying aggressively hierarchical planning strategy differs from our knowledge-based planning approach, which has previously been used to connect robot vision and grasping with symbolic action [20]. In more recent work, Kaelbling and Lozano-Pérez [14] present a belief space planner which models probability distributions over states, making it robust against uncertainties. In contrast, our choice of symbolic planner is more geared towards structured environments with incomplete and discrete information. In both approaches, sensing actions may be formulated as preconditions for manipulation, and need not be hard-coded as tasks themselves.

TABLE I
REMOVE n OBJECTS: SYMBOLIC DOMAIN DEFINITION

```
symbols
  types: object;
  predicates: isRemoved/1;
  constants: object bottle1, bottle2 ...

action remove(?o : object)
  preconds:
    K(!isRemoved(?o)) &
    forallK (?p : object)
      ( K(isRemoved(?p)) |
        !extern(graspMotionCollides(?o, ?p)) )
  effects:
    add(Kf, isRemoved(?o))

goal: forallK (?o : object)
  (K(isRemoved(?o)))
```

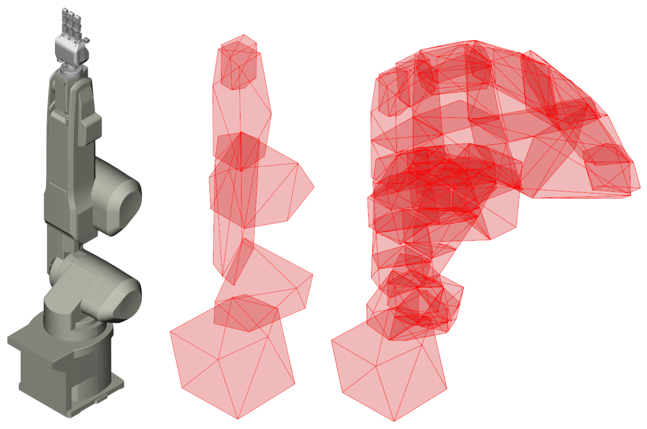

Fig. 2. Convex decomposition allows an efficient approximation of swept volumes [9]. A typical robot mesh has 10^6 vertices and is non-convex (left). Convex decomposition [16] simplifies this to 6 convex bodies with 10 vertices each (centre), allowing a typical swept volume description with only 40 convex polyhedra, totalling 400 vertices (right).

III. APPROACH

We will demonstrate our approach with several examples, among them the very simple REMOVE n OBJECTS scenario defined in Table I, which involves clearing objects from a table. In order to pick up and remove an object, collisions with other objects need to be avoided. A successful plan must therefore include `remove` actions in an appropriate order (see Figure 4). To represent these manipulation actions, we automatically decompose robot and object geometry models into sets of convex polyhedra, which can be prepared off-line. During planning, the evaluation of kinematic and geometric queries (e.g., `graspMotionCollides` in the `remove` action) requires the calculation of motion paths, swept volumes, and collision checking. Symbolic planning is performed by the PKS planner, which invokes the geometric reasoner as needed, to evaluate preconditions and effects. We discuss the algorithms and components behind this process in greater detail below.

A. Convex Decomposition of Volumes

In our KVP approach, it is crucial to efficiently represent arbitrary objects and robot manipulators as sets of convex polyhedra. In the general case, however, decomposing a non-convex polyhedron into a small (or even minimal) set of convex polyhedra is a challenging problem. Although the problem of minimal, exact decomposition is known to be NP-hard, Mamou and Ghorbel [16] recently proposed an approximate algorithm that is sufficiently efficient and precise for practical instances. This approach shows better approximation results than existing algorithms, both with respect to approximation errors as well as the number of decomposed convex polyhedra.

In principle, their algorithm hierarchically segments the non-convex polyhedron on its dual graph by half-edge decimation. The segmentation is guided by a weighted cost function, trying to minimize concavity and an aspect ratio measure they define as the squared perimeter divided by the area of a given mesh, adjusted by a constant factor to yield one in the case of a disk. The cost function is weighed such that the aspect ratio guides the first few iterations of the algorithm, quickly simplifying the mesh. After that, the simplification is mostly lead by the concavity measure, which they define as the maximum distance of mesh points projected onto the convex hull of that mesh, measured in surface normal direction.

As indicated in Figure 2, Mamou and Ghorbel's algorithm produces a concise set of convex bodies, which are well suited for efficient collision detection. For instance, a typical six-axes robot manipulator can be approximated by 6 to 10 convex polyhedra, totalling no more than 100 vertices. The swept volume of a typical motion of such a robot will simplify to no more than 20–100 convex polyhedra.

B. Efficient Swept Volume Computation

The representation of a robot's volume as a small set of convex polyhedra allows us to efficiently compute the swept volume of a robot motion. In the static case, the volume of the robot geometry R is given by the union of n convex hulls of vertices $R = \cup_n \operatorname{conv}(V_n)$. More generally, we can also efficiently approximate the swept volume SV of a robot geometry R along a configuration space path Q. In order to show the quadratic convergence of the swept volume approximation, we first consider the motion of a single convex polyhedron, which is given by the vertices V. Let r_{max} be the maximum distance of a vertex to the first axis, summing up all link lengths and the distance to the last axis. It is a well-known fact that a point on a single link of length r rotating by an angle $q < \pi/2$ will deviate from the chord (straight line from start to end) by $\epsilon = r\cos(q/2)$ [23]. For general serial kinematics of multiple revolute joints, Baginski [1] describes an upper bound for the deviation of the path of a point from the chord of that path:

$$\epsilon \leq r_{max}\left(1 - \cos\left(\frac{\sum_i |q_i|}{2}\right)\right). \tag{1}$$

Intuitively, this bound is tight when all link lengths but the last one tend to zero and all joints rotate in the same direction around the same axis. ([22] describes a similar bound by roughly approximating path segments as screw motion.) In our implementation, we calculate the bound for each link and obtain tighter bounds especially for the first few joints.

Using the second order series expansion of the cosine

$$\cos(q) \geq 1 - \left(q^2/2\right) \tag{2}$$

we can approximate Eq. 1 as a quadratic function of the angular step size Δq for our swept volume approximation:

$$\epsilon \leq r_{max} \frac{\left(\sum_i |\Delta q_i|\right)^2}{8}. \tag{3}$$

Choosing an angular step size $\Delta q = \sqrt{8\epsilon / r_{max}}$, we can therefore generate swept volumes at a desired precision ϵ, and at a quadratic convergence rate. To construct the swept volume, we compute the convex hull conv for each convex polyhedron sampling the path Q as $q(i)$ at angular distances Δq and applying the forward kinematic transformation FK, as implicitly done in [22, 23]:

$$\mathrm{SV}(R, Q) = \bigcup_n \bigcup_i \mathrm{conv}\left(\mathrm{FK}(q(i+1))V_n \cup \mathrm{FK}(q(i))V_n\right). \tag{4}$$

Denoting $|Q|$ as the length of the path in joint space, it follows that $\mathcal{O}(n\ \ |Q| / \sqrt{\epsilon})$ convex polyhedra are needed to represent the swept volume of a robot motion at a precision of ϵ, with each one having at most $2|V|$ vertices. This result implies that doubling the number of sampling points of the path Q will quadruple the precision of the approximation. In order to construct a conservative (superset) swept volume, ϵ-enlarged models of R may be computed off-line and—provided R is a conservative approximation of the real robot geometry—all computed swept volumes are conservative and, most importantly, false negatives in collision checking are avoided.

For collision checking, we use the Bullet Physics Library[1] implementation of the Gilbert-Johnson-Keerthi (GJK) algorithm [11], which has been shown to detect collisions between two convex polyhedra at a computational complexity linear in the total number of involved vertices. For an environment of m convex polyhedra, a collision check with a swept robot volume can be performed in $\mathcal{O}(nm\,|Q| / \sqrt{\epsilon})$ time. However, for many practical problems, the computation can be leveraged by fast broad-phase algorithms and may be considerably quicker.

C. Planning with Knowledge and Sensing (PKS)

Symbolic planning in KVP relies on the general-purpose PKS planner [18, 19], 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 representations based on belief states.

PKS is based on a generalization of STRIPS [7]. 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 and has a fixed, formal interpretation in a modal logic 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 of its 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 two additional databases, K_x and LCW, which aren't used in this paper.)

PKS knowledge states can be queried in three different ways. First, simple knowledge assertions can be tested with a query of the form $K(\phi)$, 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 database representation and query mechanism, 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. For instance, Table I shows the definition of a PKS action `remove(?o)`, which has the effect of clearing an object `?o` from a table.

PKS also has the ability to model sensing actions that return information about the state of the world, an idea that is important for robot task planning [10]. 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 are executed in the world. Using these databases, PKS can reason about the possible outcomes of such actions during plan construction, by generating plans with contingencies. For instance, the `senseWeight(?o)` action in Table III is an action of a sensing action that tests whether an object `?o` is spillable or not.

[1]http://bulletphysics.org (accessed May 15, 2013)

TABLE II
BIMANUAL SCENARIO: EXAMPLE ACTIONS

```
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))
```

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. PKS can also build plans with contingencies by considering 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 then continues along each branch until the goal conditions (a set of queries) are satisfied.

A second feature of PKS which is key to the KVP approach is its ability to integrate externally-defined procedures (e.g., from support libraries) with its internal reasoning mechanisms. While the idea of transferring reasoning to external processes is not a new idea [5, 6, 4], the introduction of such techniques in PKS is a more recent extension. A special keyword, `extern` provides an interface to this facility, where an expression `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 PKS's knowledge state, 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 perform additional tests on this value, or include it in its knowledge base. For instance, the `remove` action in Table I uses an `extern` call to invoke path planning and collision checking in the evaluation of `graspMotionCollides`. An `extern` call can also be directed to cache its return value for efficiency. As a result, PKS's core reasoning capabilities can be augmented by the addition of motion planning, collision detection, and other special purpose robotics libraries.

IV. EVALUATION

We now demonstrate and evaluate the efficiency of our approach with three scenarios. In the REMOVE n OBJECTS scenario, a single manipulator is used to remove n objects from a table while avoiding collisions. This scenario can readily readily be generalised to multiple manipulators, as shown in the BIMANUAL scenario. As a third example, we give the FORCE SENSING scenario, which used a sensing action that can be modelled by the knowledge-level planner.

For the REMOVE n OBJECTS scenario defined in Table I, we evaluate the performance of the planner with respect to the number of objects n, shown in Table IV. For this simple scenario, the planning time seems almost linear in the number of objects. In general, planning time is lower than or within the same order of magnitude as typical execution on a robot would take. Even though the number of collision tests may be higher than quadratic in the worst case, it must be noted that collisions of convex polyhedra can be checked very efficiently and collision checking only amounts to a negligible fraction of the total planning time for the numbers observed. Swept volume generation time mostly depends on the number of objects grasped, and slightly on the location of those grasps. In the worst case, only one of the objects can be picked up at a time. Even though this will increase the number of collision checks, influence on the total planning time is minor. As a demonstration, a sample task with $n = 3$ bottles was executed on a real robot, as depicted in Figure 4.

The BIMANUAL scenario shows that pick-and-place actions can be planned for arbitrary numbers of manipulators, denoting the respective robot `?r` as an argument for each manipulation action in the symbolic domain, as shown in Table II. It is worth noting that multiple manipulators may lead to non-

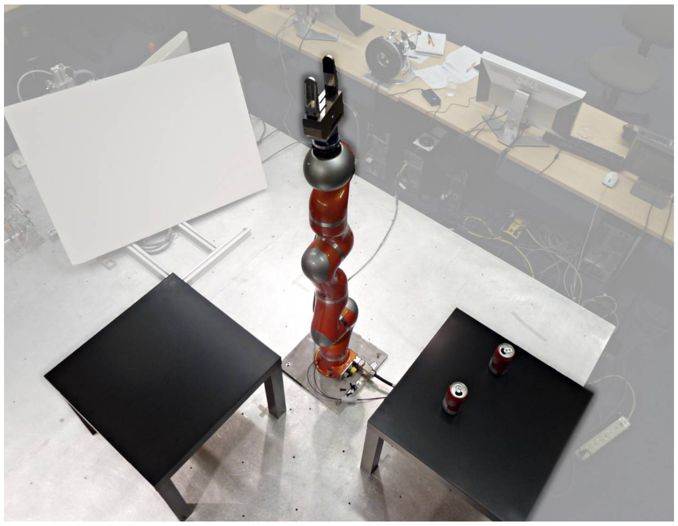

Fig. 3. In the FORCE SENSING scenario, a compliant robot manipulator can sense if beverage containers are filled by weighing them, and holds them upright while moving to prevent spilling, unless they are known to be completely empty or not opened. [10]

TABLE III
FORCE SENSING SCENARIO: EXAMPLE ACTIONS

```
action senseWeight(?o:object)
    preconds:
        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))
```

TABLE IV
EVALUATION OF THE REMOVE n OBJECTS SCENARIO

Number of Objects	Total Planning Time [s]	Inverse Kinematics [s]	Path Planning [s]	Swept Volume Generation [s]	Number of Triangles Generated	Number of Convex Bodies Generated	Number of Collision Tests	Collision Testing [s]
Objects at random locations								
2	2.77	0.35	0.43	1.99	1936	121	1	.00006
3	5.49	0.73	0.40	4.36	3168	198	4	.00021
4	8.42	1.03	1.06	6.33	4400	275	6	.00010
5	13.40	1.72	1.55	10.13	8976	561	22	.00049
6	14.60	1.77	2.69	10.14	8976	561	18	.00030
7	18.52	1.99	4.35	12.18	10912	682	22	.00035
8	19.15	2.32	2.71	14.12	12672	792	35	.00050
9	22.21	2.77	3.04	16.40	13376	836	70	.00088
10	24.33	2.99	3.10	18.23	15312	957	94	.00135
15	40.32	4.91	5.63	29.78	25872	1617	314	.00306
20	56.70	6.60	10.16	39.93	34848	2178	530	.00601
"Worst case": Objects in a line, only 1 of n can be picked up								
2	2.66	0.34	0.35	1.97	1936	121	1	.00004
3	5.56	0.69	0.78	4.09	4224	264	3	.00009
4	8.30	0.99	1.13	6.18	6160	385	6	.00012
5	10.93	1.33	1.45	8.15	8096	506	10	.00018
6	13.33	1.63	1.78	9.92	9680	605	15	.00029
7	18.97	2.25	2.89	13.83	13552	847	36	.00059
8	21.81	2.57	3.50	15.74	15488	968	44	.00071
9	25.21	2.97	4.54	17.70	17424	1089	112	.00140
10	27.85	3.23	5.12	19.50	19360	1210	125	.00170
15	42.01	4.89	7.73	29.36	28336	1771	544	.00581
20	56.81	6.72	8.75	41.32	36256	2266	1257	.01227

trivial behaviour: in order to move bottles from the right side of the bar to the goal location (Figure 1), one arm needs to move them to a location that the second arm can reach—an intermediate location to "pass it on" to a different manipulator. A previous implementation of this scenario, demonstrated on a two-manipulator robot setup, is described in [9, 8].

The FORCE SENSING scenario (Figure 3) illustrates the use of sensing actions. (A more detailed account of this domain is given in [10].) Table III shows the definitions of the sensing action `senseWeight` and the manipulation action `transferUpright`. When grasping a beverage container `?o`, the robot can sense its weight and will come to know whether it can be spilled or not, adding this knowledge to PKS's K_w database. The planner will then generate a contingent plan with binary branches, each of which account for one possible outcome of this knowledge at execution time.

V. CONCLUSION AND FUTURE WORK

In this paper, we extend our "knowledge of volumes" approach to robot task planning (KVP) and its representation of volumes as sets of convex polyhedra. In particular, we present a conservative ϵ-precise algorithm for swept volume computation of sets of convex polyhedra with quadratic convergence and discuss the efficiency of this representation in collision detection and, more generally, in pick-and-place tasks. We further demonstrate the effectiveness of KVP in several scenarios, including those involving multiple manipulators.

As future work, we plan to generalize our approach to mobile manipulation and investigate tighter bounds on Δq with respect to the kinematics, possibly allowing even more efficient swept volume generation.

ACKNOWLEDGEMENTS

The authors would like to thank Quirin Fischer for his help with the implementation and evaluation. This research was supported in part by the European Union's 7th Framework Programme through the JAMES project (Joint Action for Multimodal Embodied Social Systems, grant no. 270435).[2]

REFERENCES

[1] Boris Baginski. Efficient dynamic collision detection using expanded geometry models. In *Intelligent Robots and Systems (IROS). Proc. IEEE/RSJ International Conference on*, volume 3, pages 1714–1720, 1997.

[2] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *Intl. Journal of Robotics Research*, 28(1):104–126, 2009.

[3] C.H. 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.

[4] 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)*, pages 114–121, 2009.

[5] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *The Semantic Web: Research and Applications*, pages 273–287, 2006.

[6] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *International Conference on Robotics and Automation*, pages 4575–4581, 2011.

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

[8] Mary Ellen Foster, Andre Gaschler, Manuel Giuliani, Amy Isard, Maria Pateraki, and Ronald 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 May 20, 2013)

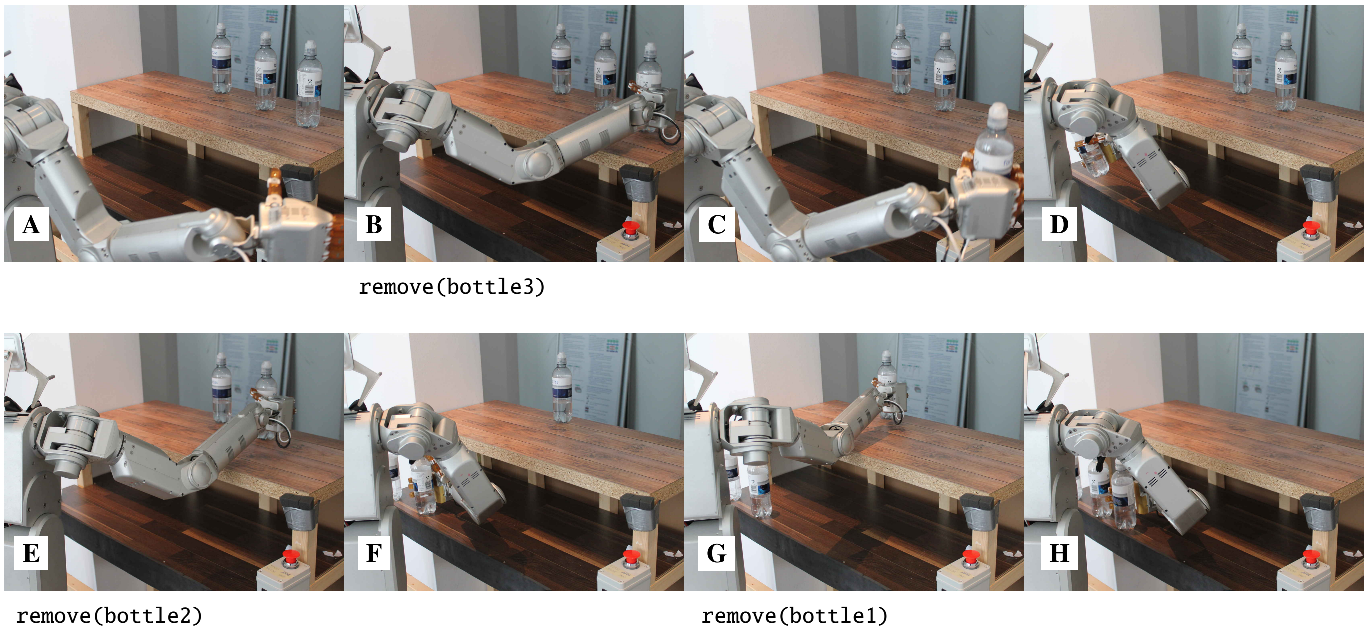

Fig. 4. REMOVE n BOTTLES scenario: The solution of the task in Table I was implemented and tested on a Meka Robotics H2. In this scenario, the robot can remove only the rightmost bottle in order to avoid collisions.

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

[10] Andre Gaschler, Ronald P. A. Petrick, Torsten Kröger, Alois Knoll, and Oussama 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.

[11] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of*, 4(2):193–203, 1988.

[12] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 30(6):678–698, 2011.

[13] L.P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477, 2011.

[14] L.P. 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] H. Kress-Gazit and G.J. Pappas. Automatically synthesizing a planning and control subsystem for the DARPA Urban Challenge. In *Automation Science and Engineering (CASE). IEEE International Conference on*, pages 766–771, 2008.

[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)*, pages 3501–3504, 2009.

[17] N.J. Nilsson. Shakey The Robot. Technical Report 323, AI Center, SRI International, April 1984.

[18] 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)*, pages 212–221, 2002.

[19] 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)*, pages 2–11, 2004.

[20] Ronald Petrick, Dirk Kraft, Norbert Krüger, and Mark 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*, pages 58–65, 2009.

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

[22] John Schulman, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. *Robotics Science and Systems (RSS)*, 2013. Accepted.

[23] Patrick G Xavier. Implicit convex-hull distance of finite-screw-swept volumes. In *Robotics and Automation (ICRA). IEEE International Conference on*, volume 1, pages 847–854, 2002.