Extending the Knowledge of Volumes Approach to Robot Task Planning with Efficient Geometric Predicates

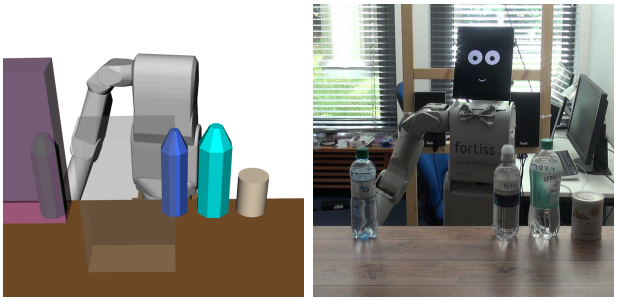
Andre Gaschler, Ingmar Kessler, Ronald P. A. Petrick and Alois Knoll

***Abstract*— For robots to solve hard tasks in real-world manufacturing and service contexts, they need to reason about both symbolic and geometric preconditions, and the effects of complex actions. We use an existing Knowledge of Volumes approach to robot task planning (KVP), which facilitates hybrid planning with symbolic actions and continuous-valued robot and object motion, and make two important additions to this approach: (i) new geometric predicates are added for complex object manipulation planning, and (ii) all geometric queries—such as collision and inclusion of objects and swept volumes—are implemented with a single-sided, bounded approximation, which calculates efficient and safe robot motion paths. Our task planning framework is evaluated in multiple scenarios, using concise and generic scenario definitions.**

I. Introduction

One of the main motivations for robot task planning is to allow intelligent robots to solve real, complex tasks in service and industry. While substantial progress has been made in recent years in designing the necessary components needed for such a framework—path planning, grasp planning, trajectory generation, and symbolic reasoning—and powerful algorithms for addressing these individual problems are available, the generic robot task planning problem remains challenging. One reason for this inherent challenge is the interdependence of symbolic and geometric reasoning: a symbolic action description (such as grasping an object) may have complex geometric preconditions, including different types of collision avoidance, and certain symbolic effects (such as placing an object on a table) may only be generated by very careful geometric choices. This implies that only trivial instances of the robot task planning problem can be solved by separating symbolic planning from subsequent motion planning of the symbolic actions; real problems can only be solved by adopting a hybrid approach.

In this work, we present two important extensions to the existing "knowledge of volumes" (KVP) approach to robot task planning [1], [2]. First, we define a set of geometric predicates in order to support generic solutions to complex manipulation problems. In addition to the previously described predicates needed for collision-free picking and placing [2], we define new predicates for inclusion checking and placing objects on support surfaces. Using these primitive queries, considerably more complex tasks can be solved; for instance, definitions of container objects such as drawer or cabinets can be specified on an abstract semantic level, enabling solution strategies such as stacking flat objects to arise from symbolic action definitions instead of through preprogramming. Second, we formulate all geometric queries as single-sided, bounded approximations with controllable precision ε. The boundedness ensures that collision and non-inclusion will always be recognized, while the less critical cases of non-collision and inclusion are approximated up to a parameter ε [1]. This type of approximation allows both safe and efficient generation of complex manipulation plans, and is formulated for the complete set of geometric predicates.



(a) Planning scenario with objects and locations (b) Evaluation in a humanoid robot setup

Fig. 1: Robot task planning provides a generic solution to complex manipulation tasks. When space is limited, the robot can stack objects where possible—a solution strategy that arises purely from symbolic and geometric planning.

In general, KVP follows two main strategies. On the symbolic side, the planner operates on the *knowledge level*, and can reason with discrete uncertainty. This allows a very clean formulation of knowledge gain and loss, information-gathering sensing actions, and plans with contingencies, as described in [3], [4], [5]. On the geometric side, data representation and predicate evaluation is centred on geometric volumes, representing all objects, robots, and even swept volumes of robot motion as sets of convex polyhedra [6]. This strategy is combined with efficient ε-precise geometric queries [1], taking a different route from and avoiding some of the issues of many sampling-based task planners.

II. Related Work

Automatic planning has been used for autonomous robot control since its very early days, for instance in the Shakey system [7]. While early approaches separated symbolic planning from geometric motion planning, it was recognised that

A. Gaschler and I. Kessler 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, Scotland, United Kingdom.

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

more complex problems intrinsically require a hybrid approach. One of the first hybrid robot task planners, aSyMov, was presented by Gravot, Cambon, and Alami in 2003 [8]. More recently, robot task planning has become a very active research area, with approaches stemming from conventional, sampling-based motion planning [9], [10], symbolic planning invoking motion-planning functions [11], [8], and probabilistic preimage back-chaining [12].

A typical approach to robot task planning is to evaluate symbolic actions in a forward-chaining manner, sampling geometric choices, and backtracking on failure, including geometric infeasibility. For instance, the aSyMov task planner [8] mentioned above consists of a symbolic planner that follows several heuristics to guide the geometric search. Symbolic and geometric searches are interleaved, with backtracking in both layers, and probabilistic roadmaps are created for all combinations of robot manipulators and objects to represent the search space. Dornhege et al. [11] add robotics-specific functions to a symbolic planner through a semantic attachment interface. Srivastava et al. [13] solve scenarios with large numbers of movable objects, provided that symbolic explanations for all failures in the geometric search are available, which are fed back to the symbolic search. Contrary to the above search schemes, Kaelbling and Lozano-Pérez's hierarchical task and motion planner "in the now" [14] performs an aggressively hierarchical, back-chaining search, whose solution is meant for interleaved execution and plan refinement. While their search technique differs from ours, we borrow their notion of continuous geometry, represent robot motions as swept volumes, and formulate similar geometric collision predicates. In addition, we develop the continuous geometry approach further, devise additional predicates, and provide an efficient and safe bounded approximation scheme (Section III). In more recent work, Kaelbling and Lozano-Pérez generalize their search space to probability distributions over states and present a belief-space hierarchical planner (BHPN) [12].

In contrast to many of the above works, our KVP approach is among the first to use a general-purpose planning system, the PKS planner [3], [4], for generic robot task planning, and the first to provide a set of efficient algorithms for fast, single-sided approximate geometric predicate evaluation.

III. Approach

Robot task planning requires techniques from multiple areas of research. In the following section we discuss our approach, beginning with the underlying symbolic planner PKS (Section III-A), followed by a description of the efficient geometric predicates (Section III-B), and finally an evaluation in an illustrating example scenario (Section IV).

In general, robot task planning requires a definition of the available actions, whose execution is subject to a number of preconditions, and which change the known world state according to a certain set of effects. A precondition or an effect may be purely symbolic, or geometric. Geometric preconditions and effects contain *geometric predicates*, whose evaluation includes queries to the kinematic and geometric models of the scenario, and may contain updates to the geometric state. For this, the task planning problem contains kinematic and geometric models of all robots and objects of the scenario, movable or static. The solution of a task planning problem is a sequence of actions whose preconditions are satisfied and whose subsequent execution produces a series of effects that brings about a knowledge state that satisfies the goal condition of the problem.

A. Planning with Knowledge and Sensing (PKS)

The underlying symbolic planner used in this work is PKS (Planning with Knowledge and Sensing) [3], [4], a contingent planner that builds plans using incomplete information and sensing actions. PKS works at the *knowledge level* by reasoning about how the planner's knowledge state changes due to action. Unlike planners that reason with possible worlds or belief states, PKS works directly with formulae representing its knowledge state. These formulae are stored in a set of databases, each of which models a particular type of knowledge. The main PKS databases we use include:

$\mathbf{K_f}$: This database is like a STRIPS database except that both positive and negative facts are stored and the closed world assumption is not applied. K_f can include any ground literal ℓ, where $\ell \in K_f$ means "the planner knows ℓ."

$\mathbf{K_w}$: This database models the plan-time effects of sensing actions with binary outcomes. $\phi \in K_w$ means that at plan time the planner either "knows ϕ or knows $\neg\phi$," and that at execution time this disjunction will be resolved.

$\mathbf{K_v}$: This database stores function values that will become known at execution, such as the effects of sensing actions that return constants. K_v can contain any unnested function term f, where $f \in K_v$ means that at plan time the planner "knows the value of f." At execution time, the planner will have definite information about f's value. Thus, K_v terms can act as *run-time variables* [15] in plans.

PKS databases are inspected using *primitive queries* that ask simple questions about PKS's knowledge. Basic knowledge assertions are tested with a query $K(\phi)$ which asks: "is a formula ϕ true?" A query $K_w(\phi)$ asks whether ϕ is known to be true or known to be false. A query $K_v(t)$ asks "is the value of function t known?" The negation of the above queries can also be used.

PKS actions are defined by *preconditions* that query the knowledge state, and *effects* that update the state. Preconditions are simply a list of primitive queries. Effects are described by STRIPS-style "add" and "delete" operations that modify individual databases. E.g., *add*(K_f, ϕ) adds ϕ to K_f, and *del*(K_w, ϕ) removes ϕ from K_w.

PKS builds plans by forward-chaining search: if an action's preconditions are satisfied in a knowledge state, then its effects can be applied to produce a new knowledge state. PKS can also build contingent plans by considering its K_w and K_v knowledge: a branch is introduced for each possible outcome of this knowledge, producing more certain knowledge states. Planning continues from each new knowledge state until the *goal*—a list of primitive queries–is satisfied.

PKS also provides a mechanism for invoking externally-defined procedures (e.g., special purpose libraries) during plan generation, `extern`$(proc(\vec{x}))$ [16], an idea similar to that of semantic attachments [17], [18], [19]. In this case, `extern` is a special keyword indicating that control should be transferred to an external procedure with the name *proc*. $\vec{x}$ is a list of parameters that should be passed to *proc*. In general, $\vec{x}$ can contain any symbols defined in PKS's knowledge state, providing a link between the planner and any externally-defined geometric predicates for motion planning and collision detection. Using this facility to reason about geometry at the symbolic level allows us to solve problems which may be difficult to model directly at either the motion planning or symbolic planning level alone.

B. Geometric Predicates

In previous versions of KVP [1], [2], the geometric predicates covered collision checking and inverse kinematics. In particular, the *isColliding(?o : object, ?p : object)* predicate checked for collisions between two given objects, and the *isCollidingMotion(?o : object, ?p : object)* predicate checked whether the robot motion to pick up a certain object would collide with another. In the following, we present additional geometric predicates to define more complex scenarios.

The *isInside(?o : object, ?l : location)* predicate reports whether the volume of an object is fully contained by the volume given by a certain location. Typically, it is used to define goal regions, rather than the discrete goal locations in [2]. More generally, geometric inclusion allows container objects such as boxes, cabinets, and drawers to be defined abstractly. For efficiency, we require the volume given by the location to be convex; the evaluation of the predicate then reduces to a simple check whether all vertices of the mesh of the object are inside the location region.

The *isOnSupportSurface(?o : object)* predicate tells whether an object rests solidly on a horizontal surface, as opposed to placement in empty space or on the edge of a table. This check is performed automatically based on the models of the object and the environment and does not need to be defined in the domain definition. We apply a simple geometric heuristic to evaluate this predicates efficiently, as it is frequently called in many manipulation task planning scenarios. In our heuristic, the predicate extends short line segments vertically down from all vertices at the bottom of the object and checks that the object itself is collision-free but that all of the line segments are not.

The *isReachable(?o : object)* predicate tells whether the robot can move to pick up an object without any collisions. This predicate is based on existing solutions for the inverse kinematics and motion planning problems provided by the *Robotics Library (RL)*[1]. To improve efficiency the inverse kinematics results are cached, which is especially effective when the inverse kinematics is only applied to a subset of the robot's reachable space, e.g., a tabletop. Furthermore, geometric heuristics are applied so that full path planning often becomes unnecessary during the planning phase.

The *freeSpaceFound(?o : object, ?l : location)* predicate tells whether an object can be put down in a location by combining the three predicates above. This predicate works as a suggester function for finding a valid geometric position for an object, at which all three other predicates are fulfilled. This is done by sampling search positions from a search area defined by the volume of a location. The sampling process is space-filling and therefore complete for the quadtree and random search strategies. For comparison, the trivial linear search strategy searches a subset of the sampling positions given a resolution parameter. The origin and direction for the linear search and the quadtree search are automatically determined depending on the orientation of the location relative to the robot, and the type of gripper used by the robot. These search strategies only determine the horizontal x and y coordinates as shown in Fig. 2 and a downward raycast is applied for finding the vertical z coordinate. This predicate enables new types of scenarios to be defined since objects may not only be moved to predefined geometric positions, but also intelligently to anywhere within a defined area.

More details on the implementation of these geometric predicates are described by Kessler in [20].

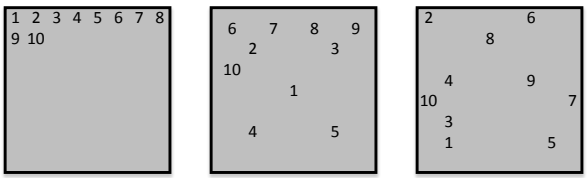


Fig. 2: From left to right, examples for the first ten sampled search positions for a linear search, quadtree search and random search.

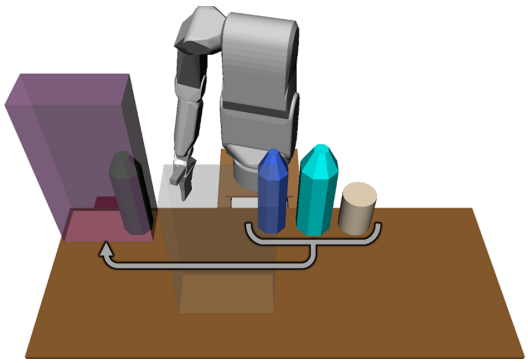
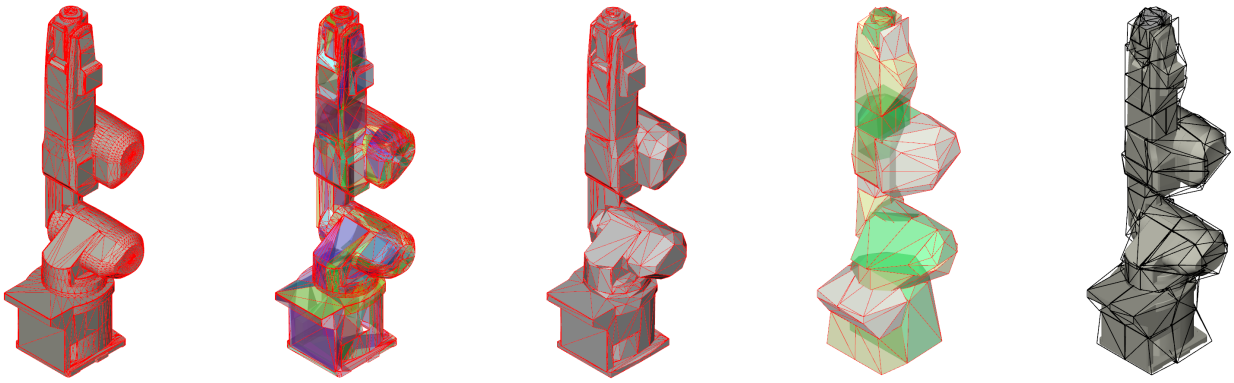


Fig. 3: The geometric world at the beginning of the MOVE AND STACK scenario.

C. Efficient Geometric Predicate Evaluation

Besides the availability of the new geometric predicates formulated in the previous section, which allow complex task planning problems to be defined and solved, it is of great importance that these predicates are implemented efficiently. In robot task planning, geometric predicates are evaluated many

[1] `http://www.roboticslibrary.org/`



(a) Original CAD Model (b) Exact Convex Decomposition (c) Bounding Mesh (d) Bounding Convex Decomposition (e) Overlay on Original Model

Fig. 4: Novel bounding mesh and convex decomposition algorithms can handle a typical, non-convex robot mesh with 10^6 vertices (a). Exact decomposition would generate overly high numbers of polyhedra (b), which are inefficient. Our new bounding mesh approximation [1] reduces the number of vertices to 10^3 at an error $\varepsilon < 0.05\,\mathrm{m}$ (c). Then, convex decomposition can simplify this mesh to ≈10 convex polyhedra with 20 vertices each (d, e). [1, p. 10]

times in a recursive search, and greater efficiency allows more complex scenarios to be solved. In our KVP approach, we propose a single-sided, bounded ε-precise approximation for all geometric queries, named *bounded geometric predicates* [1]. A bounded geometric predicate has the following properties. On one side, the approximation is guaranteed to be exact. For instance, the *isInside* predicate will always return false if an object is not completely contained by a certain area, and the *freeSpaceFound* predicate will always fail if a collision occurs. This behavior allows safe motion planning and is important to many robotics applications. On the other side, the approximation is bounded by a parameter ε. In case of the *freeSpaceFound* predicate, it is allowed to fail if two bodies almost collide with a distance closer than ε. Conversely, the *isInside* predicate is only guaranteed to recognize inclusions when the distance to the inner body is at least ε.

The bounded geometric predicates can be evaluated very efficiently when bounded convex decompositions of all objects and robot models in the task planning scenario are available from a preprocessing step. This ε-bounded convex decomposition is generated through several steps, illustrated in Fig. 4. A typical geometric model of a robot manipulator may contain at the order of 10^6 vertices (see Fig. 4a). An exact convex decomposition is neither computationally efficient, nor would it reduce this number of vertices (see Fig. 4b). Our novel bounding mesh algorithm [1], however, generates a bounding mesh, a simpler mesh that includes the original, at a distance no more than ε (Fig. 4c), effectively reducing the number of vertices to the order of 10^3. Then, hierarchical convex decomposition routines [21] can decompose this bounding mesh into a small bounding set of convex bodies (Fig. 4d). Being convex, the resulting bodies can be efficiently used in all primitive collision and inclusion checks using the Gilbert/Johnson/Keerthi (GJK) algorithm [22], leveraging the efficiency of all described geometric predicates.

TABLE II: Evaluation of the REMOVE n OBJECTS scenario, where n objects must be removed from a table while avoiding collisions. Geometric queries such as collisions are made efficient by the swept volume representation as sets of convex bodies. [6, p. 5]

Number of Objects n	Total Time [s]	Inverse Kinematics [s]	Path Planning [s]	Swept Volume Generation [s]	Collision Checking [s]	Number of Geometric Precondition Queries	Number of Geometric Effect Evaluations	Number of Symbolic Actions in Solution
Objects at random locations								
2	12.41	0.01	12.21	0.19	.00001	3	2	2
3	40.35	0.01	39.72	0.62	.00034	6	3	3
4	39.48	0.01	38.74	0.70	.00111	10	4	4
5	102.30	0.03	101.46	0.81	.00146	15	5	5
10	143.67	0.04	141.74	1.84	.01712	120	15	10
20	356.98	0.10	354.01	2.81	.04442	320	64	20
Objects in a line, only 1 of n can be picked up								
2	12.43	0.01	12.23	0.19	.00001	3	2	2
3	39.87	0.01	39.44	0.42	.00003	6	3	3
4	99.46	0.02	99.01	0.43	.00004	10	4	4
5	158.70	0.02	158.26	0.42	.00005	15	5	5
10	351.77	0.02	350.67	1.08	.00270	55	10	10
20	355.16	0.12	352.43	2.57	.02974	210	20	20

IV. EVALUATION

The MOVE AND STACK scenario was chosen for testing and evaluating the geometric predicates introduced in Section III-B since it is not a pure pick-and-place scenario, but rather allows generic stacking of objects. The combined symbolic and geometric planner solves the scenario with the help of the *isOnSupportSurface* predicate, which allows for generic and abstract stacking that does not need to be preprogrammed. For this predicate the top of the can is semantically the same as the tabletop and the only domain

TABLE I: Performance evaluation of the MOVE AND STACK scenario and three simpler variants A, B, and C. Abbreviations: bB := blueBottle, gB := greenBottle, tB := turquoiseBottle

Scenario Variant	A	B	C	MOVE AND STACK
Goal Conditions	K(isInside(bB, depot)) ∧ K(¬isInside(gB, tray))	K(isInside(bB, depot)) ∧ K(isInside(tB, tray)) ∧ K(¬isInside(gB, tray))	K(isInside(tB, tray)) ∧ K(isInside(can, tray))	K(isInside(bB, tray)) ∧ K(isInside(tB, tray)) ∧ K(isInside(can, tray))
Number of Actions	6	8	10	12
Number of Geometric World States	26	55	79	92
Total Time [s]	0.64	1.13	2.26	3.15
thereof Planning Time [s]	0.09	0.39	0.78	1.13
thereof Path Planning [s]	0.55	0.74	1.48	2.02
Path Optimization Time [s]	8.27	11.35	23.53	30.62
(Path Planning if performed at every symbolic step [s])	4.83	15.50	31.41	43.37
Inverse Kinematics [s]	0.027	0.104	0.110	0.121
(Inverse Kinematics without Caching [s])	0.068	0.289	0.628	0.918
Collision Checking in Planning Time [s]	0.023	0.159	0.410	0.630
isInside Number of Calls	21	161	418	698
isInside Total Time [s]	0.003	0.017	0.044	0.072
isOnSupportSurface Number of Calls	13	153	410	690
isOnSupportSurface Total Time [s]	0.000	0.009	0.024	0.040
isReachable Number of Calls	38	168	357	535
isReachable Total Time [s]	0.063	0.240	0.387	0.528
freeSpaceFound Number of Calls	14	42	86	128
freeSpaceFound Total Time [s]	0.045	0.293	0.595	0.879

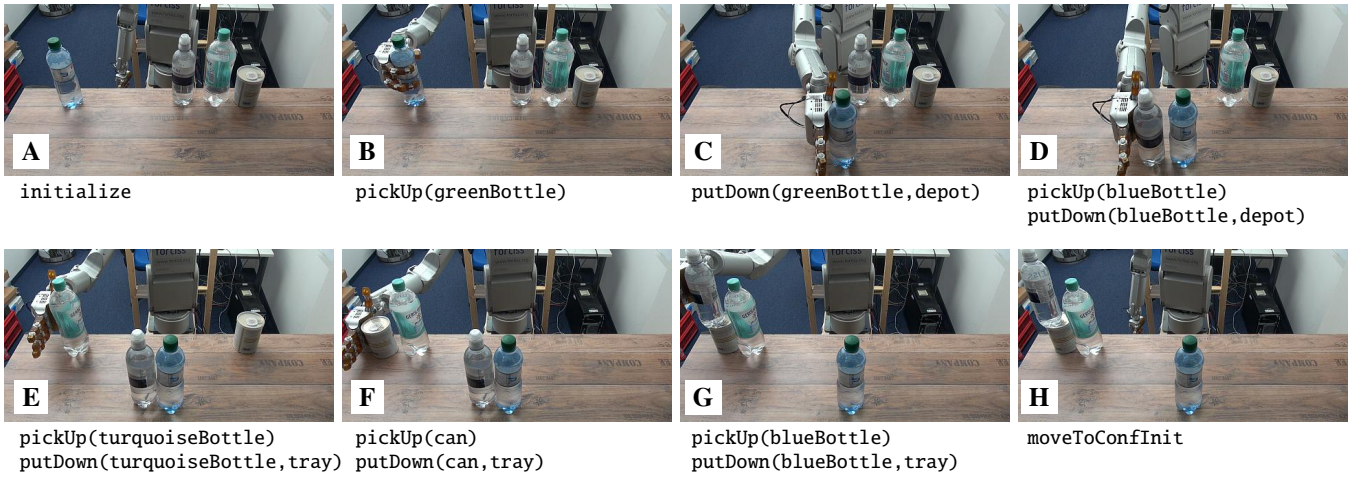


Fig. 5: Solution for the MOVE AND STACK scenario. At the beginning, the small green bottle has to be moved to another location to clear space for later actions (image **B**) and the small blue bottle is temporarily moved to the *depot* location as well so that the next object in the row can be accessed. Next, the large turquoise bottle and the can are moved to the *tray* location. Afterwards, the small blue bottle is automatically stacked on top of the can inside the *tray* location since the three objects do not fit inside the *tray* location side by side (image **G**). A video is available at `http://youtu.be/XGSMoI_BRFw`.

definition necessary was to design the predicate so that all horizontal surfaces are suitable for resting an object on them.

Fig. 3 shows the geometric world at the beginning of the scenario. The grey arrow illustrates the symbolic goal of moving all three objects on the right into the small purple-transparent *tray* location on the left. The green and blue bottles are small enough to fit on top of the can but the turquoise bottle is not. The green bottle is simply in the way and the white-transparent *depot* location is simply an alternative location for storing objects. The scenario is complicated by the small size of the locations and the fact that the objects frequently act as obstacles when performing actions on other objects. The combined symbolic and geometric planner solved this scenario and executed the resulting plan on the robot in the real environment in 103 seconds, as shown and explained in Fig. 5. Table I shows the performance in the MOVE AND STACK scenario and several similar scenarios with different goal conditions. The data illustrates that the total time is well below the execution time and that the

Fig. 6: REMOVE n BOTTLES scenario: In this scenario, the robot can remove only the rightmost of $n = 3$ bottles in order to avoid collisions. [6, p. 6]

geometric predicates are efficient with their computational time less than the path planning time.

A. *Remove n Objects Scenario*

In order to show that general types of scenarios can be solved by our task planning approach, we briefly discuss the REMOVE n OBJECTS scenario, which was previously described in [6]. In this second scenario, the robot is supposed to remove n objects from a table while avoiding collisions, so this problem can be studied quantitatively. As shown in the results in Table II, the planning time scales reasonably with the number of objects. While collisions are common for higher numbers of objects, most of these geometric predicate calls can be cached, and a goal-directed search can solve this problem rather efficiently. A demonstration with $n = 3$ objects was performed on the real robot, shown in Figure 6.

V. CONCLUSION AND FUTURE WORK

In this paper, we extend the existing "knowledge of volumes" (KVP) approach to robot task planning, adding additional geometric predicates, and improving the efficiency of all collision and inclusion queries by single-sided bounded approximations. We demonstrate that these new predicates allow generic definitions of support surfaces and enable non-trivial task planning solutions, such as stacking objects in confined workspaces.

ACKNOWLEDGEMENTS

This research was supported by the European Union through the projects JAMES under grant agreement no. 270435, and SMErobotics under grant agreement no. 287787. The authors would like to thank Quirin Fischer for his help with the evaluation.

REFERENCES

[1] A. Gaschler, R. P. A. Petrick, O. Khatib, and A. Knoll, "A knowledge of volumes approach to robot task planning," 2015, submitted.

[2] A. Gaschler, R. P. A. Petrick, M. Giuliani, M. Rickert, and A. Knoll, "KVP: A Knowledge of Volumes Approach to Robot Task Planning," in *IEEE/RSJ Intl Conf on Intelligent Robots and Systems (IROS)*, November 2013, pp. 202–208.

[3] R. P. A. 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.

[4] ——, "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.

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

[6] 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.

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

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

[9] E. Plaku and G. D. Hager, "Sampling-based motion planning with symbolic, geometric, and differential constraints," in *IEEE Intl Conf on Robotics and Automation (ICRA)*, 2010, pp. 5002–5008.

[10] J. L. Barry, "Manipulation with Diverse Actions," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[11] 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.

[12] L. P. Kaelbling and T. Lozano-Pérez, "Integrated Task and Motion Planning in Belief Space," *International Journal of Robotics Research*, vol. 32, no. 9–10, pp. 1194–1227, 2013.

[13] S. Srivastava, E. Fang, R. Lorenzo, R. Chitnis, S. Russell, and P. Abbeel, "Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer," in *Robotics and Automation (ICRA). IEEE International Conference on*, 2014.

[14] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Intl Conf on Robotics and Automation (ICRA)*, 2011, pp. 1470–1477.

[15] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson, "An Approach to Planning with Incomplete Information," in *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR)*, 1992, pp. 115–125.

[16] R. P. A. Petrick and A. Gaschler, "Extending Knowledge-Level Contingent Planning to Robot Task Planning," in *Workshop on Planning and Robotics (PlanRob) at the Intl Conf on Automated Planning and Scheduling*, June 2014, pp. 157–165.

[17] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits, "Effective integration of declarative rules with external evaluations for semantic-web reasoning," in *The Semantic Web: Research and Applications*, 2006, pp. 273–287.

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

[19] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Intl. Conf. on Robotics and Automation*, 2011, pp. 4575–4581.

[20] I. Kessler, "Robot Task Planning with Efficient Geometric Predicates," Master's thesis, Technische Universität München, Munich, Germany, August 2014.

[21] 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.

[22] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.