Identifying Good Poses When Doing Your Household Chores: Creation and Exploitation of Inverse Surface Reachability Maps

Andreas Hertle Bernhard Nebel

***Abstract*— In current approaches to combined task and motion planning, usually symbolic planning and sampling based motion-planning are integrated. One problem is here to come up with good samples. We address the problem of identifying useful poses for a robot close to working surfaces such as tables or shelves. Our approach is based on reachability inversion which answers the question: where should the robot be located in order to reach a certain object? We extend the concept from point-based objects to flat polygonal surfaces in order to enable the robot to have a a good grasping position for many objects. Our approach allows to quickly sample multiple distinct poses for the robot from an prior computed distribution. Further we show how sampling from an inverse reachability distribution can be integrated into a CTAMP system.**

I. Introduction

Much research effort is devoted to bring autonomous robots to our households. There is a huge potential: robots could clean up the living space, assist the owner in the kitchen and clean up the table after meals. Such support is in particular helpful, when the owner is not capable of performing these tasks. While special purpose robots like autonomous vacuum cleaners are a success story, so far general purpose personal household robots are not commercially available. Besides unsolved challenges in the fields of perception and mechanical articulation, the integration of diverse skills and their coordination is one of the obstacles.

For instance consider the problem a robot has to solve in order to clean a table. First, the robot would need to get close to the table in order to observe the state of the table and to identify possible objects on it. Depending on the observations, the robot might need to remove dirty dishes before wiping the table. Most likely, the robot needs to relocate multiple times in order to reach all the dishes or the dirty parts of the table.

Symbolic planning is one possible approach to solve such problems on an abstract level. It allows the robot to explore how to apply its skills in order to reach a certain goal. The skills of the robot are encoded as symbolic actions, where the complex geometry is abstracted away. Because of this abstraction, symbolic planners can explore the huge state space resulting from applying all actions efficiently. However robots move through and interact with a three dimensional world. It can not be expected to generate only executable plans when geometry is abstracted away. A trade off between symbolic efficiency and geometric feasibility is necessary. This is known as the *combined task and motion planning* problem. In *CTAMP* symbolic and geometric planning is interleaved: symbolic planning leverages the strength of abstraction to explore efficiently towards a goal while geometric planning ensures that the produced plan are geometrically correct and executable on the robot. In order to gear the combined planning process into the right direction, it is necessary to restrict the number of poses a robot can take to those that are *useful*.

In this paper we address the problem of identifying useful poses for a robot close to working surfaces such as tables or shelves. Our approach is based on reachability inversion which answers the question: where should the robot be located in order to reach a certain object? We extend the concept from point-based objects to flat polygonal surfaces in order to enable the robot to have a good grasping position for many objects. Our approach allows to quickly sample multiple distinct poses for the robot from an prior computed distribution. Further we show how sampling from an inverse reachability distribution can be integrated into a *CTAMP* system.

Fig. 1: The robot needs to determine a suitable pose close to the table in order to interact with objects on the table.

II. Motion and Task Planning

As mentioned, when a robot has to plan and act in a complex environment, it is necessary to plan its different tasks as well as to plan its motions. So, if we consider this combined planning problem, it is planning in the product space of these two planning problems. Given that both of these problems are already highly intractable, the combination is, of course, worse.

The authors are with the University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany.

This work was supported by the PACMAN project within the HYBRIS research group (NE 623/13-1). This work was also supported by the DFG grant EXC1086 BrainLinks-BrainTools to the University of Freiburg, Germany.

A. Semantic Attachments

A number of approaches have been developed to deal with this problem. One approach is based on a generic interface between a symbolic planner and an embed planner, such as a motion or manipulation planner. The symbolic planner is extended by so-called *semantic attachments* [1], [2], which provide the interface to the embedded planner.

A classical symbolic planning task is represented as a finite number of objects of certain type and Boolean facts about objects and relations between objects. A *state* represents a particular valuation for every fact in the planning task. *Actions* define the transition between states. Before an *action* can be applied its preconditions must be satisfied by the current state. The application of an action produces the successor state according to the specification of its effects on the facts. The planning process succeeds when an action sequence from the initial state to a state fulfilling the goal condition is found.

For example, the initial state of the example in the introduction could be described by stating which objects are on the table, where actions are picking up objects, putting down objects and cleaning the table. This all does not include any geometric information, despite the abstract relationship of being standing on the table.

Of course, for developing feasible plans, the geometric information has also to be taken into account. This is done by using different forms of semantic attachments that refer to continuous variables, describing the exact geometrical position of all objects, the pose of the robot as well as the pose of the affecter. There are three different forms of such attachments. First there are the *condition checker attachment* that check the continuous state space for certain conditions, e.g., whether an object is graspable. Second, there are the *effect attachment* that effectively change parts of the continuous state space. This could be, for instance, a new pose after a movement. Both of these attachments are completely deterministic and deliver the same result if evaluated in the same state (symbolic and continuous).

B. Sampling Poses

While planning, the symbolic planner branches over all possible actions in order to look for a plan that leads to a goal state. In order to be complete, it also needs to branch over all (reasonable) values in the continuous state space, e.g., poses of the robots and grasp positions. For this purpose, we have introduced a *grounding attachment* that returns samples from the continuous state space. So, we do not have to discretized the world before starting the planning process, but can interleave the sampling with the planning process. The important point to note here is that this kind of branching over samples is interleaved with branching over possible actions, leading to a potentially infinitely branching search tree.

Of course, one is interested in reasonable samples in order to steer the search for a plan into the direction where a goal state can be expected. And precisely at this point, inverse reachability maps come into play.

III. Surface Inverse Reachability

We extend the concept of reachability inversion from point-based objects as introduced by Vahrenkamp et al. [3] to flat polygonal surfaces like tables or shelves. A *point-centered inverse reachability map* is referred to as *point map* in order to contrast it with surface inverse reachability map or *surface map* for short. The intent of a *surface map* is to quickly sample poses for a robot in order to interact with the surface or (previously unknown) objects on it.

A robot pose is defined as $p = \langle x, y, z, \theta \rangle$ where x,y,z are the spatial coordinates and θ corresponds to the planar orientation of the robot. The reference frame of the pose can be chosen for the specific robot: For wheeled robots it might be advantageous use the base frame. For legged robots the torso frame might be more practical. The *point map* assigns each robot pose p a *reachability index* R_P, a normalized measure of quality:

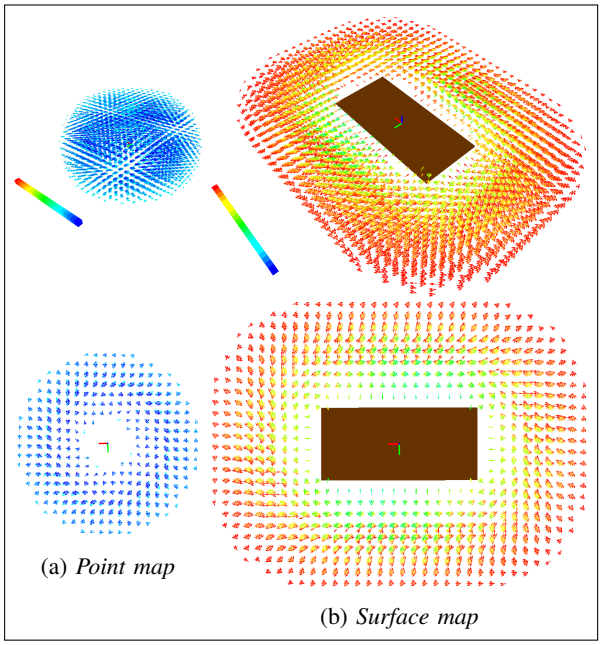
$$p \mapsto R_P, R_P \in [0, 1]. \tag{1}$$

It can be interpreted as how many different grasp solutions are valid from that particular pose. Our *surface map* analogously provides a surface reachability index R_S for each pose p around the given polygon. R_S intuitively corresponds to the percentage of surface area that the robot can reach from each pose.

A. Construction of a Surface Map

To construct a *surface map* we need the polygon of the surface and a *point map* for the robot manipulator and consequently the *surface map* is only valid for that polygon and manipulator. The idea is to evaluate the *point map* on various sampled points inside the surface polygon and accumulate the quality measures. To simplify the accumulation process we construct the *surface map* with the same resolution as the *point map* and align the orientation of both maps, so when we move the *point map* to a sampled point we can transform between *point map* and *surface map* with simple vector additions. Furthermore we can restrict the sampled points to a grid with the same resolution as the *point map*, since sampling finer grained does not provide more information. Before we can transfer the quality measures form the *point map* to the *surface map* each robot pose has to be validated. The robot must be able to assume the pose without colliding with the surface polygon. Also a collision free inverse kinematic solution to reach the sampled point must exist. Without the IK verification the *surface map* would incorrectly count instances where the robot is able to reach the target from below through the surface polygon. During validation it is also possible to add robot specific constraints to filter invalid poses and reduce the computational effort. For instance the z coordinates could be restricted based on the height of the robot and the working surface.

The accumulation procedure is as follows: We sample a point inside the polygon and shift the origin of the *point map* to that point. Then we transfer the quality measures R_P for any valid pose to the *surface map*. Quality measures from previously sampled points are added to the new ones.



(a) *Point map* (b) *Surface map*

Fig. 2: On the left the *point map* (a) that was used to construct the *surface map* (b) is shown. The bottom shows horizontal slices through both maps. Red colors indicate a low, blue colors a high reachability.

Finally we normalize the magnitude of the accumulated quality measures as shown in equation (2):

$$R_S = \frac{1}{n} \sum_{i=1}^{n} R_{Pi}, \tag{2}$$

where n is the number of sampled points inside the surface polygon.

So far we only considered robots with one manipulator. However it is easy to combine multiple *surface maps*. Let's assume we have a robot with two manipulators and corresponding *surface maps*. If we take the maximum of $R_S = \max(R_{S1}, R_{S2})$ for a pose, the resulting reachability will indicate how good any manipulator can reach the surface. If on the other hand we take the minimum $R_S = \min(R_{S1}, R_{S2})$ we know how good both manipulators reach the working surface, which is useful for two-handed robot skills. It is possible to combine *surface maps* offline and produce a *surface map* for both manipulators. However since the R_S lookup is highly efficient the maps can also be kept separately for increased flexibility.

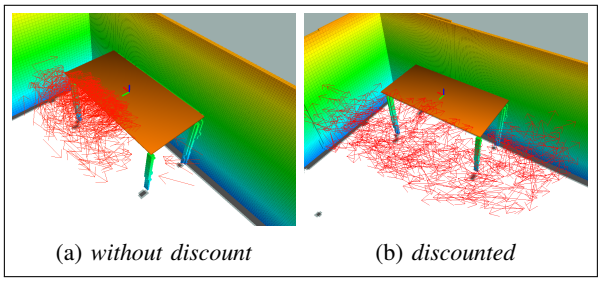
B. Sampling from a surface map

When the robot needs a good pose close to a working surface we are not interested in exactly the pose with the best R_S, we only want a reasonably good pose.

To obtain a robot pose with a reasonable R_S we sample random (continuous) pose candidates inside the bounding box of the *surface map*. Each candidate is put trough a number of test to ensure validity. The candidate can be discarded if:

1) the reachability index $R_S = 0$ as it is impossible to reach the surface from this candidate;
2) it is located outside operational environment, e.g. determined by a navigation cost map;
3) the robot would collide with the environment or other objects, determined by a full state collision check.

Each test is computationally more expensive than the previous to eliminate invalid candidates with as little effort as possible. Once we have a certain number N of valid candidates we commit to the one with the highest R_S.



(a) *without discount* (b) *discounted*

Fig. 3: Sampling 500 consecutive poses without (a) and with (b) reachability discount. The discount pushes consecutive poses apart and helps covering the whole range of possible poses.

Successfully completing a task could require the robot to move various poses around the surface. When repeatedly sampling poses from a *surface map* it is crucial not to receive the same "best" pose. Instead we would like them to be spread out so that the whole surface can be approached. Therefore we keep track of previously sampled poses and discount the R_S of poses close to it. We use the normalized Euclidean distance since it allows to weight dimensions individually and to compare linear distances and angular distances. The computation of the discount factor is shown in equation (3)

$$w = \prod_{p_i \in P} \min\left(1, \sqrt{(c - p_i)^T S^{-1} (c - p_i)}\right) \tag{3}$$

where P is the set of previously sampled poses and c is the current pose candidate. S is a normal distribution that defines the closeness measure; we use a diagonal matrix with $S = (0.5, 0.5, 0.5, \pi/4)$. The difference between taking previous poses into account and simple consecutive sampling is shown in figure 3.

C. Surface Maps for CTAMP

With our *surface maps* we implemented two different *semantic attachments*.

When the robot plans to interact with a working surface or objects on it, we need to ensure the pose in that future state is close enough to the surface. Thus we have a *condition attachment* to query the reachability for a pose, a simple lookup in the *surface map* for that surface. This condition is added to actions like *inspect-surface* or *pickup-object*. Should the retrieved reachability be $R_S = 0$, the action can

be discarded before more expensive computations would be performed.

Furthermore we implemented a *grounding attachment* that allows to create new poses for the robot by sampling from a *surface map*. Without the ability of adding new poses during planning process, it has to be decided beforehand how many poses the robot will need to solve a task and where those are located. If too few poses are given the task might become unsolvable. If too many poses are given the complexity of the planning task is increased unnecessarily. With this attachment the planning system is able to gradually increase the number of poses should the previously sampled poses be insufficient to solve the task.

IV. Related Work

Our work is embedded in the area of combined task and motion planning as well as in the area of precomputing reachability maps, inverse reachability maps and similar concepts.

A. Combined Task and Motion Planning

In the area of combined task and motion planning, there are a number of related approaches. However, similar to the the one sketched here, most of them are based on sampling-based robot motion planning integrated with symbolic planning [4], [5], [6], [7]. They differ on how the integration is performed, and on what kind of planning strategies are used, but they all contain as one important part the sampling of poses. A somewhat different approach is the one by Toussaint, who considers the CTAMP problem as an optimization problem [8].

B. Sampling the Pose Space

Selecting good samples is, as mentioned above, one important subtask. Here, we find a number of interesting approaches that inspired our work.

Zacharias et al. [9] introduced a representation of a robots arm kinematic capabilities in its workspace: the so-called *capability map*. This map is built offline to support online queries. With the help of such maps, a robot can easily compute how to reach a target configurations in 3D space. A scaled down version of this map is the *reachability map* that does not contain the solution for the inverse kinematic query but simply a quality measure of how good an object can be grasped.

Vahrenkamp et al. [3], [10] presented an approach to invert a workspace representation to build a so-called *inverse reachability map* with the goal to select appropriate robot base positions for executing grasp tasks. In their work, they use an extended manipulability measurement as a quality index capturing the robots maneuverability in the workspace. In order to find suitable base poses for object manipulation, the inverse reachability map is placed at the targets pose. In a next step, a robots base pose is sampled and followed by an inverse kinematic query to find a joint configuration that reaches the goal position while avoiding collisions. Unfortunately, they assume that the pose of how to grasp the object is already known, which in real-world applications is not always the case. Similarly to Vahrenkamp et al., Burget and Bennewitz [11] also employ inverse reachability maps. However, they use them on a legged platform instead of a wheeled one.

Another related approach is the one by Stulp et al. [12]. They try to identify so-called *action related places*. These places are not single base poses, but are collections of positions with each one possessing a probability for the success of a given manipulation action. Further, they use a transformational planner in order to come up with a combined task and manipulation task.

V. Experimental Evaluation

We evaluate our *surface map*s and their integration into a *CTAMP* system with a PR2 robot simulated in Gazebo. The PR2 robot features two 7-DOF manipulators, a laser range finder mounted on the base for localization and obstacle detection and a Kinect sensor mounted on the head. The robot operates in a small room with two tables and a number of small objects on them, like cans and cups.

The robot is controlled by a planning system with the Temporal Fast Downward planner with semantic attachments at its core. The planner is embedded in a continual planning loop that interfaces the planner with the robot system: The state of the robot and the environment is observed and converted to a symbolic representation. When a plan is found the symbolic actions trigger corresponding robotic skills. After each action is executed the state is re-estimated to verify that the remaining plan continues to be valid. Should an unexpected event occur, like an action not producing a desired effect or the discovery of new objects, re-planning is started.

We modeled a the robot's capabilities in our mobile-pick-and-place PDDL domain. The *move-robot* action allows the robot to relocate to a another location. With the *inspect-surface* action the robot observes a table with the Kinect camera to detect and recognize objects on the table. The *pickup-object* action allows the robot to grasp previously recognized objects with either manipulator. Once grasped the object can be held to the camera in order to *inspect-object*. Finally, a grasped object can be placed on a table with the *putdown-object* action. Each action where the robot moves is accompanied by a *cost attachment* and an *effect attachment*. The cost attachments compute more realistic expected durations for the actions and at the same time verify the geometric correctness. The effect attachments propagate the geometric information like object coordinates to subsequent states. We previously implemented this system on a PR2 robot in the Tidy-Up Robot project [13]. However, without *surface map*s the robot was only able navigate to predefined poses, a shortcoming we want to address in this work.

A. Data Structure and Generation

The data structure behind *surface map*s is a custom extension of OctoMap [14]. When retrieving the reachability for

a certain pose, the coordinates x, y, z and θ are discretized according to a resolution of the *surface map*. x, y and z signify which voxel needs to be found and the voxel stores reachability values for θ if $R_S > 0$. If no entry is retrieved the reachability is implicitly 0. The complexity of the lookup operation for a random poses is in $O(d) + O(\log n)$, where d is the tree depth and n is the number of discrete θ angles.

The computational effort to generate a *surface map* strongly depends on the chosen resolution. For instance consider a rectangular table surface with the dimensions $1.4m$ times $0.65m$. When using coarse resolution of $0.1m$ and $\frac{2\pi}{16}$ radians we get $14 * 6 = 84$ sample points on the surface to evaluate. The constructed *surface map* contains reachability values for approximately 1 million robot poses. It was generated within approximately 2 hours on a 3 GHz processor. For the finer resolution of $0.05m$ the number of samples on the table increases to $28 * 13 = 364$ with reachability information for a total of 20 million poses. Computation for the finer resolution took approximately 2.5 days. The resulting OctoMap file size is merely 3 MB.

For the following experiments we use the finer resolution of $0.05m$ with a an angular resolution of $\frac{2\pi}{16}$. When sampling new poses we return the best of 40 candidates. Subsequent sampling of new poses is discounted according to distances to previous poses when closer than $0.5m$ or $\pi/4$ rad.

B. Plan Viability with Sampled Poses

In this experiment the robot has the task to inspect every object found on a table. To inspect the objects the robot has to pick them up and hold them in front of the camera. Initially the robot is located away from the table and has no knowledge of the objects on the table. Only when moving to a table and inspecting it, the robot perceives the objects and can interact with them.

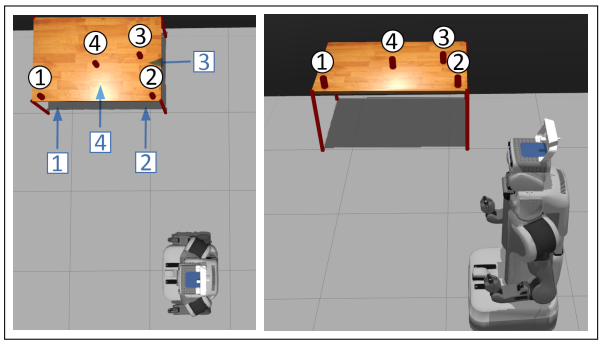


Fig. 4: Initial situation for the inspect objects task. The blue arrows indicate the predefined poses.

Between 1 and 4 objects are added on top of the table. First, the robot is given a number of predefined poses that have been recorded manually. Second, we sample poses from a *surface map*. The results are shown in table I.

The total execution time between individual runs can vary to a large degree, depending on which poses the robot chooses to approach first. With the predefined poses some tasks can not be solved, because some objects can not be reached from any of the given poses. Overall the robot can solve the task in a comparable amount of time when sampling poses. Thus when sampling from *surface map*s the system works with less given information and solves more tasks.

TABLE I: Total execution time in minutes with predefined poses and poses sampled form a *surface map*.

	1 object	2 objects	3 objects	4 objects
1 pose	7.9	-	-	-
2 poses	8.6	19.5	-	-
3 poses	10.7	31.4	43.0	-
4 poses	8.6	16.9	34.5	87.9
sampling	15.3	36.4	37.0	64.1

C. Influence on Plan Quality

In the second experiment we asses the influence of sampled poses on the overall plan quality. The task is again to inspect every object found on a table. Initially the robot is located away from the table and has no knowledge of the objects. The experiment is repeated seven times with predefined poses and another seven times with sampled poses. We recorded the total execution time until the task was solved, the amount of time spent planning and how many re-planning attempts were necessary. Furthermore we recorded how often the robot would move to a different pose and how many non-move actions were executed during each run. The results are shown in table II. The total execution time in minutes is denoted T and the time spent re-planning t_p. Furthermore, the table shows the number of re-planning processes, the number of move actions and the number of other actions executed by the robot while solving the task.

TABLE II: Solving the task of inspecting three objects with predefined poses (top) and sampled poses (bottom).

	1	2	3	4	5	6	7	average
T [min]	43	35	44	36	35	37	37	38.0
t_p [min]	10.3	5.1	13.5	5.2	3.7	5.8	5.8	7.0
# re-plan	9	5	13	6	4	7	7	7.3
# move	5	3	3	3	4	5	4	3.9
# other	27	29	34	26	27	28	27	28.3
	1	2	3	4	5	6	7	average
T [min]	37	25	32	42	31	29	28	32.0
t_p [min]	9.3	4.0	10.8	11.8	5.0	5.23	5.9	7.42
# re-plan	8	6	15	16	6	7	7	9.3
# move	4	1	1	1	3	2	3	2.1
# other	24	26	30	39	28	27	26	28.6

When looking at the runs with predefined poses the first and the third runs are noticeable for their long execution time. During run 1 the robot moved five times; clearly sub-optimal with only four predefined poses. The planning system produces sub-optimal plans fast and then tries to improve the solution. In this case it did not manage to optimize away the extra move action. For the third run a lot of time is spent re-planning. A detailed analysis of the action log revealed that one manipulation action was repeatedly

failing for unknown reasons, sending the system into re-planning.

When examining the results of the runs with sampled poses in the lower half of table II, the runs 1 and 4 have the longest execution time. It seems run 1 was unlucky when sampling poses and had to relocate three more times before the task was solved. Run 4 and to a lesser extend run 3 were again suffering from execution failures as indicated by the increased number of re-planning steps and manipulation actions. However, the frequent failing of actions might also result from sampling a pose from where it is difficult to grasp a certain object. On the other hand in some runs the sampling produced exceptionally good poses so that the robot could inspect all objects without relocating. Overall during the runs with sampled poses fewer move actions were necessary and the total execution time was on average 5 minutes lower than with predefined poses.

VI. Conclusions

In this work we addressed the problem of choosing suitable robot poses close to tables, shelves or other working surfaces. Initially the robot does not know if or where objects are located on the working surface. So our approach derives suitable robot poses based on the shape of the working surface and the robots manipulation capabilities.

The capability of a robot can be encoded in a *reachability map*, that allows to decide quickly which objects are in reach given the robot pose. Such a map can be inverted to provide suitable poses for the robot given the object coordinates. We showed how the concept of *inverse reachability maps* can be extended to find good poses for the robot close to flat polygonal surfaces. Such *surface map*s are computed offline and are afterwards available for efficient online queries.

We demonstrate how *surface map*s can be integrated into a symbolic planning system. We implemented two *semantic attachments*. A *condition attachment* quickly verifies whether the robot is positioned correctly in order to interact with the surface. A *grounding attachment* adds a pose to the planning state allowing the planner to evaluate alternative robot poses. The planning system is able to adopt the number of poses as necessitated by the task at hand.

In our experiments we examined the influence of sampled poses on task solving. When sampling poses from *surface map*s the robot relies less on a priori given expert knowledge. In some cases the system could even find a solution, where predefined poses were insufficient to solve the task. On average the task could be solved faster and with fewer robot movements.

References

[1] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós, "A comparison of SLAM algorithms based on a graph of relations," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, 2009, pp. 2089–2095. [Online]. Available: http://dx.doi.org/10.1109/IROS.2009.5354691

[2] A. Hertle, C. Dornhege, T. Keller, and B. Nebel, "Planning with semantic attachments: An object-oriented view," in *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, 2012, pp. 402–407. [Online]. Available: http://dx.doi.org/10.3233/978-1-61499-098-7-402

[3] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013, pp. 1970–1975. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2013.6630839

[4] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. J. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 639–646. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6906922

[5] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011, pp. 1470–1477. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2011.5980391

[6] 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 *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011, pp. 4575–4581. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2011.5980160

[7] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, 2015, pp. 6366–6373. [Online]. Available: http://dx.doi.org/10.1109/IROS.2015.7354287

[8] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 1930–1936. [Online]. Available: http://ijcai.org/Abstract/15/274

[9] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA*, 2007, pp. 3229–3236. [Online]. Available: http://dx.doi.org/10.1109/IROS.2007.4399105

[10] N. Vahrenkamp and T. Asfour, "Representing the robot's workspace through constrained manipulability analysis," *Auton. Robots*, vol. 38, no. 1, pp. 17–30, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-014-9394-z

[11] F. Burget and M. Bennewitz, "Stance selection for humanoid grasping tasks by inverse reachability maps," in *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, 2015, pp. 5669–5674. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2015.7139993

[12] F. Stulp, A. Fedrizzi, and M. Beetz, "Action-related place-based mobile manipulation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, 2009, pp. 3115–3120. [Online]. Available: http://dx.doi.org/10.1109/IROS.2009.5354281

[13] C. Dornhege and A. Hertle, "Integrated symbolic planning in the tidyup-robot project," in *AAAI Spring Symposium - Designing Intelligent Robots: Reintegrating AI II*, Mar. 2013. [Online]. Available: http://www.informatik.uni-freiburg.de/ ki/papers/dornhege-hertle-aaai13ss.pdf

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com