A two-level approach for intricate manipulation planning

Mihai Pomârlan[1] and Ioan A. Şucan [2]

[1]Universitatea Politehnica Timisoara, facultatea ETC
[2]Willow Garage

***Abstract*—In the past few years several research efforts have aimed to produce robots capable of operating in environments inhabited by humans. This requirement places some tough constraints on a robot; among them, the ability to solve manipulation tasks that come as second nature to people. Manipulation problems are difficult because they present narrow passages (often, the kinds of possible or useful maneuvers are included in a very small subset of all possible maneuvers) as well as very high dimensional configuration spaces. Manipulation tasks often require both arms to be used, and the grasp on a manipulated object matters. Furthermore, the grasp may need to be changed along the execution of the task. To make planning for such problems tractable, we propose a two-level planning architecture inspired by the way humans tackle such problems: plan a motion for the object first, then the sequence of manipulation actions. This approach can alleviate both the narrow passage and the high-dimensionality problem, as well as offer the possibility for human assisted motion planning. We offer an intuitive justification for the method and validate it experimentally.**

***Index Terms*—robot motion planning, manipulation planning, sparse roadmaps, human assisted planning**

I. Introduction

Sampling-based planners [1], [2] have become the dominant approach in most practical planning applications for systems with many degrees of freedom (e.g., robotic arms), because of their ability to usually find solutions in a short amount of time [3]. Compared to other methods such as cell decomposition planners [1], they make fewer assumptions on the configuration space and are therefore more widely applicable, and do not require the exact computation of the configuration space — only the ability to sample it.

Avoiding the computation of an explicit representation of the free part of the configuration space usually offers a speed advantage because planning environments are often "simple", with large free spaces. However, sampling-based planners are notoriously slow at handling problems with narrow passages (e.g., [4]–[6]). This is a fundamental consequence of sampling. A narrow passage is an area of free space of small volume, and poor visibility from other portions of the free space. Therefore, if a solution requires traversing a narrow passage, then the planner must place samples in the passage, which is unlikely due to the region's small volume.

Meanwhile, problems of manipulation may present narrow passages. Objects often need to be manipulated in close proximity to another, and may need to pass through or around one another. Consider for example placing a key in, or removing it from, a keychain, or hanging a coat in a wardrobe. Day to day life presents several such problems of (dis)entangling objects. The circumstances in which these appear are also varied, so simple static approaches (e.g., as done for industrial robots programming [7]) may not always be adequate. The quintessential disentangling problem is the puzzle, and by construction these are made to be hard for humans too. While rather artificial, they easily showcase, in an exaggerated form, the kinds of problems described in this paper.

Fig. 1. Several (dis)entangling tasks of various levels of difficulty: using a coat hanger, removing a key from a keychain, a metal cast puzzle. While the puzzle is specifically designed to try a human's patience, it is simply an exaggerated version of everyday manipulation tasks.

Apart from the narrow passages, manipulation problems can also be difficult because of the number of degrees of freedom (DOF) involved. Often, the manipulation will have to be performed with two arms, either because there are two or more objects that must be manipulated, or because manipulating one requires changing grasps. This greatly increases the configuration space to explore for a solution, especially if the grasps themselves add even more degrees of freedom (the placement of gripper elements around the object) to consider.

If possible, planning in high-dimensional spaces should be avoided, and this appears to be what human intuition does to handle manipulation tasks. We seem to first think of how the manipulated object should move, or at least establish a few waypoints, and then think of how the arms should move and grasp.

Therefore, one of the contributions of this paper is to propose a two-level approach for complex manipulation planning: plan for the object considered as if it were capable to move on its own, then for the arms to manipulate it into following the solution trajectory. The benefit of our approach is planning now happens in the configuration space of the object, rather

[1]mihai.pomirlan@etc.upt.ro
[2]isucan@willowgarage.com

This work was partially supported by the strategic grant POSDRU 107/1.5/S/77265, inside POSDRU Romania 2007-2013 co-financed by the European Social Fund Investing in People.

than the configuration space of robot with the object. The plan we obtain for the object will constrain the space we need to search for arm and grasp planning. Rather than searching in the full space of possible motions of the manipulator, we search the subspace capable to manipulate the object towards following the previously planned trajectory. This greatly reduces dimensionality and improves efficiency. Another contribution is representing, in a compact fashion, a set of useful motions in the object configuration space as a roadmap; for example, trajectories passing through narrow passages. We can then concatenate trajectories from the roadmap into more complex plans as needed by the task. The construction of the roadmap for the object is done with human assisted motion planning, to ease identification of the narrow passages and interesting regions in configuration space. The capability to use human in the loop planning is also important in some contexts like industrial robotics, as it allows the operators to have a greater degree of control over the robot's repertoire of manipulation maneuvers.

We present a planning architecture that fits our approach, and implement it atop the MoveIt! software package [8]. We describe the method, and a test class of problems, in section III. We give some experimental results and analysis in sections IV and V.

II. Related Work

The idea of guiding planning in a high dimensional space with one in lower dimensions has appeared before, for example in [9] or [10]–[12]. The difference between [9] and this work is we do not use a discrete decomposition of workspace as the lower dimension guide, but rather a continuous motion of the manipulated object. Meanwhile, [10]–[12] concern themselves with planning for movement constrained to a certain submanifold of the workspace, either known ([10], [11]) or identified by running principal component analysis on the growth of a tree planner [12]. Our approach is applicable to motion in any subset of the workspace, and is complementary to constrained sampling approaches.

Industrial assembly problems also require precise movement and placement of objects relative to one another. However, in such applications the task is rigorously defined beforehand and the environment is tightly controlled, so the robot is simply given a script of movements to follow. There is little to no autonomy required [7]. More recently, some research has gone into developing ways by which a casual user, as opposed to a robotic line programmer, could teach a robot to perform a task [13]. The eventual goal of such research is to have the human user demonstrate a task, and have the robot follow it [14]. The focus is not on autonomy but on finding new ways to communicate a script to a robot.

Autonomous handling of complex tasks also requires task planning. Typically, task planning approaches are also two-level, however the distinction is between a symbolic level, at which the task itself is specified, and a geometric level in which the robot motion is described.

The symbolic level should allow one to reason about the task and define sequences of subtasks, in such a way that proceeding through a task plan, one will only do subtasks for which all preconditions have been satisfied. Actions and their effects are described in some formal way like STRIPS or some version of temporal logic, to allow an inference engine to find logically feasible sequences of actions to satisfy task preconditions [15]–[17].

The geometric level is the one in which motion planning happens. Satisfying geometric constraints, such as avoiding obstacles, and finding a motion to complete a subtask, is handled by this level. It provides the input to lower level control, to actually perform the robot motion. Note that a subtask sequence may be logically feasible, and geometrically infeasible (there is no way to complete one of the subtasks). Various ways for the symbolic and geometric planning levels to interact have been proposed.

One of the more studied approaches is one called 'manipulation graphs' [18], [19]. One defines subspaces of the configuration space of the robot and manipulated object, where a subspace is defined by the grasp the robot keeps on the object (one could call such subspaces CG, for continuous grasp); there is a subspace in which the robot does not grasp the object (this subspace is often referred to as CP, for continuous planning). Sampling-based planners create roadmaps for such subspaces, and two kinds of maneuvers are defined: transit maneuvers or paths, in which the manipulated object is not grasped and stays in some stable configuration; and transfer maneuvers or paths, in which the robot carries the object. Such maneuvers can be concatenated at the intersection of the CP and relevant CG subspaces. The various connections between subspaces define a manipulation graph, and task planning becomes a path search in this graph.

The manipulation graph approach has the grasp as the point of interaction between the symbolic and geometric levels [19]: grasping is a geometrical relation between the robot and object, and limits what movements the robot can now do. It is also a logical property; many subtasks would have as precondition that an object be grasped in some fashion, sometimes even limiting the allowable grasps to only a few. Switching grasps is sometimes necessary both for geometric reasons (to continue a movement) as well as logical ones (the next subtask requires it as a precondition). Manipulation graphs try to explicitly model allowed transitions between grasps [18]. As part of such a process, one often needs stable configurations for the manipulated object, where the robot can safely leave the object ungrasped.

A number of refinements on the manipulation graph idea have been proposed. Constraint satisfaction algorithms are used to find destination configurations for motion planning in a task planning context in [20]. Several parallel PRM searches are run for different subtasks in [21], to avoid having the task planner get stuck when one of the motion planner queries is infeasible. The redundancy of humanoid robots is exploited in [22], which considers the several ways a robot has to go about completing a task. For example, it may use one arm or

the other; it may use an arm motion to move an object, or employ the moving base as well. To capture such variety of action, [22] proposes a Task Motion Multigraph structure. An "aggressively hierarchical" task representation is done in [23]: the task planner commits itself to a sequence of subtasks, and splits each subtask into lower level subtasks until "atomic" subtasks, motion planner queries, are generated.

Unlike most of the previously mentioned approaches, the method proposed in this paper does not require maintaining or constructing several roadmaps for various possible grasp states. We do not sample in the full space of degrees of freedom of the robot (arms and grippers), but instead sample in the space of configurations of a rigid object and we keep one single roadmap. This single roadmap provides sufficient constraint to quickly guide path generation for the entire robot, and can be augmented with information to ease grasp selection at various stages of plan execution as well. Testing possible grasps based on an already constructed roadmap for the rigid object is faster than sampling in the full robot configuration space.

III. Problem Description

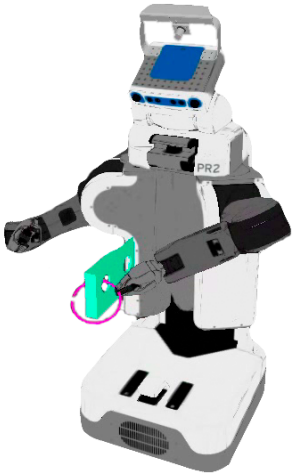

Fig. 2. The planning problem: the robot needs to take the ring from the start to some given goal configuration. The card piece is fixed in space and creates an obstacle with narrow passages.

The problem class considered as a test case in this paper is manipulating a ring piece around a card with two holes. A particular planning problem in this class requires the ring to be taken from some start to some goal state. A small arc of the ring is missing, which allows it to slide along the card and through the holes. The card is assumed to be a fixed obstacle in space and the the ring is assumed to be stable at the start and goal configurations. In our experiment, the robot doing the manipulation is a PR2. The PR2 has two 7-DOF arms, and a typical problem will require the use of both arms. Planning is done using a modified OMPL plugin [24] for the MoveIt! package [8], and the simulation is viewed in RViz[1].

[1]The default visualization tool for ROS

For this particular problem class, one needs to make use of both the robot's arms and grippers, which results in a fairly high dimensional configuration space. Furthermore, by design, there are several narrow passages to make the problem difficult. However, it is natural to consider subspaces of fewer dimensions when solving this problem, and we pursue such an approach here.

One level of planning concerns the object to be manipulated, treated as a rigid body capable to move under its own power, in the environment with obstacles (but without the robot being present, or not considered as an obstacle in any case). The purpose of this level of planning is to provide a solution path for the object to follow from start to goal.

Of course, in reality the object is not capable to move by itself. The second level of planning is then tasked with getting the robot arms to move in such a way, so they will drag the object along the solution path provided by the first level. The movements of the arms must themselves be feasible, and the feasibility constraint may require grasp changes on the object, for example when continuing to move the object using the current grasp results in a collision, or is kinematically impossible.

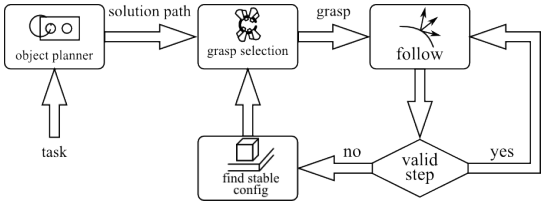

Fig. 3. The proposed method: first, find a solution assuming the manipulated object can move autonomously. Then select grasps and plan for the arms, so as to drag the object along the solution trajectory.

There are therefore two subtasks the second level of planning needs to implement. One is grasp selection and switching, and it provides the start and goal configurations for the second subtask, actual planning of arm motion. It should be noted that grasp switching is itself a task on which some intuitive constraints are imposed. Simply put, the robot should not drop the object. Or, if the robot does release the object at some point, it should leave it in a stable or otherwise predictable configuration.

Grasp selection can also be guided by a few criteria. The most obvious is a grasp needs to be feasible before it can even be considered as an option. Some kind of optimality criteria may also help refine the search. In particular, one can try to select grasps in such a way so as to minimize the number of necessary grasp switches during the solution, or, as we do here, try "promising" grasps first and backtrack if the robot encounters dead ends.

Planning first for the manipulated object as if it were capable to move on its own does not consider whether the robot will actually be capable to perform the required maneuvers to make the object follow our planned path, and as a result our approach is not probabilistically complete. However in practice this is an acceptable price to pay for the increased

efficiency of planning in a smaller dimensional space; our experiments show the planner is capable to find solutions for various problems reliably. Probabilistic completeness is also a fairly weak guarantee for practical usage, as by itself it does not indicate how quickly a planner would converge to a solution. The time a planner has allocated for finding a solution is also limited, so even a probabilistically complete planner will fail on occasion because of converging too slowly. In particular, increasing the dimensionality of the space requires more exploration steps from a planner and slows convergence to a solution when narrow passages are present, so while not probabilistically complete, a planner working in lower dimensional spaces may well converge to a solution faster in practice.

A. Planning a path for the manipulated object

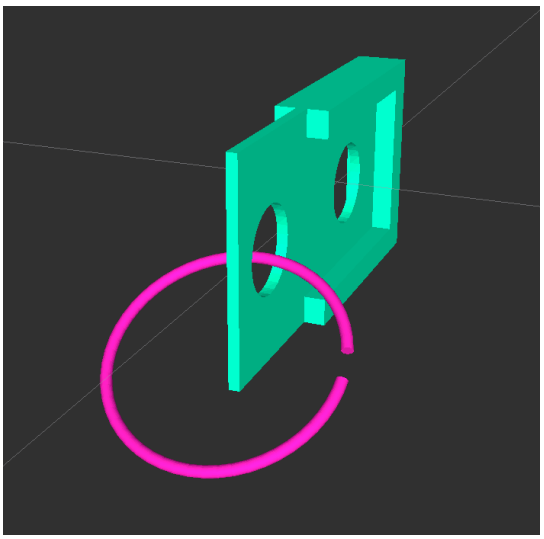

Fig. 4. Planning first considers the ring as an object capable of autonomous movement. The robot does not exist in this planning environment.

Algorithm 1 GrowRoadmap (G, q_{start}, q_{goal})

$startNear \leftarrow \text{NearestNeighbors}(G, q_{start}, r)$
$goalNear \leftarrow \text{NearestNeighbors}(G, q_{goal}, r)$
$nearVerts \leftarrow startNear \cup goalNear$
$connComps \leftarrow GetConnectedComponents(nearverts)$
if $(\emptyset = connComps)$ **or** $(1 < |connComps|)$ **then**
  $G \leftarrow G \cup \{q_{start}, q_{goal}\}$
  for $q \in startNear$ **do**
    $\text{addEdge}(G, q_{start}, q, |q_{start} - q|)$
  for $q \in goalNear$ **do**
    $\text{addEdge}(G, q, q_{goal}, |q_{goal} - q|)$
  RunPlanner(G, q_{start}, q_{goal})

The first step is to obtain a solution to the simplified motion planning, which considers the ring as a free-flying object. Note that the small missing arc, the card and the holes create several narrow passages. As a result, the single query sampling-based planners implemented in MoveIt will have a difficult time solving it. Even with a couple of minutes allowed, we found such planners fail to find a solution for problems manipulating the ring through the card. These planning problems are for a relatively small-dimensional space (the possible configurations of the ring, which has six degrees of freedom), not the space of the two manipulator arms (which has fourteen). Trying to use a single-query planner on the larger space would require even longer planning times.

On the other hand, a human can easily identify a few interesting configurations for the ring, which will guide the search for a solution. For this class of problems, humans can quickly see the narrow passages and place samples inside them. Thus, a roadmap is constructed with human assistance, to speed up planning. The roadmap construction proceeded as illustrated in algorithm 1: the user would set a start and goal configuration, then request a plan to link them. The customized OMPL planner then attempts to link the start and goal to the constructed roadmap. If they can be linked to the same connected component, then they are not interesting samples and discarded, to keep the roadmap small. However, if they cannot be linked to the same component (because the roadmap is fragmented, or empty, or simply cannot see the start or goal), then they are added to the roadmap, and the planner spends some time generating random samples in an attempt to connect start and goal.

At this stage, with the roadmap empty or small, and the user would place start and goal at interesting locations. In effect, the user grows the roadmap, not the planner. At each step, a list of maximally connected components of the roadmap is maintained. The goal is to have a roadmap with several interesting configurations of the ring around the card, which also contains only one maximal connected component.

We used 167 vertices in the roadmap. Thus prepared, the planner proved capable to solve planning queries for the ring quickly and reliably.

One thing to notice is, the strategy presented here constructs a solution path, then imposes on the robot the task to make the ring follow the path. The path may become infeasible if changes in the environment occur. However, that is not a disadvantage of our strategy; if the solution becomes infeasible, then replanning is possible, with the currently reached configuration as a starting point. Indeed, because of the presence of narrow passages, simple path deformation may not be adequate, leaving replanning as the only choice. Further, having a roadmap can greatly speed replanning for a problem with narrow passages, assuming the roadmap is rich enough to capture a wide enough variety of behaviors, while also being small enough to be fast to query.

Finally, using a precomputed roadmap is not equivalent to storing one precomputed behavior for a given task. The roadmap is capable to answer more planning queries, and it needs not be constrained to a particular environment. In our case, only the ring and the card are relevant when the roadmap is constructed. The environment can be different and include more obstacles when the roadmap is actually used. As long

as the roadmap is rich enough to contain various movements of the ring relative to the card in an otherwise empty space, it will work well in an environment with other obstacles and/or the card in a different pose. To handle such situations, one can use roadmap planners adapted for changing environments, for example lazy versions of PRM.

B. Planning for the arms

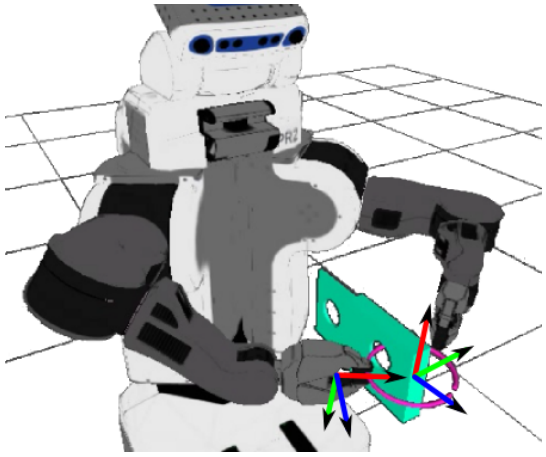

Fig. 5. Planning for the arms: gripping with manipulating arm, moving unused arm away. Making the arm drag the ring along a specified path is achieved by using the transformation between gripper and ring piece, which provides inverse kinematics targets for the arm to follow.

The planning for the robot arms consists of selecting a grasp, then using inverse kinematics to follow the path given by the ring planner. To ease this, a finite set of known grasps (twentyfour) is defined for each arm, relative to the ring. Grasp selection then is done as a greedy choice: the grasp that can be maintained the longest along the manipulation plan is picked first.

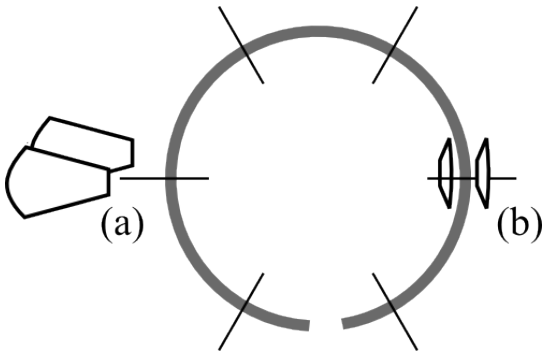

Fig. 6. A finite set of grasps is defined. Six positions along the ring are allowed for grasping, and for each position, the grasp can be sideways (gripper direction axis in the ring plane; situation (a)) or aligned (gripper direction normal to the ring plane; situation (b)). Each grasp type is further divided into two subtypes, either regular or flipped: gripper rotated 180 degrees on an axis, compared to the regular subtype.

The ring cannot be dropped by the robot, except at the start and goal points. To change grasps, we use the fact that the robot has two arms. When changing a grasp, the ring is already held by, for example, the right arm. We would then choose a grasp for the left arm, and bring it toward the ring with motion planning. Then, we pose a motion planning problem for the right arm: it needs to let go of the ring, and return to some safe position away from the solution path. The reason for this last step is obvious, but it highlights another component of the manipulation task, namely that the robot is itself an obstacle the object must be manipulated around. Therefore, some way to formulate a policy of keeping the robot out of the way, and a component to enforce the policy, are needed, even for simple cases such as tucking away an unused arm.

Once the ring is in the grasp of an arm, we use the gripper-ring transformation defined by the grasp to see the path the end effector needs to follow, so as to drag the ring as required. We use inverse kinematics to follow this path, and validate at small intervals along it. We check that obstacle collisions do not occur, the end effector pose is kinematically feasible, and it does not require the arm to do a "jump" in joint space. If any of these conditions fail, we stop following the trajectory and switch grasp.

A dead end can happen when, by following the object manipulation plan with the currently selected grasp, we arrive at a configuration which is not the goal, from which it is impossible to continue with the current grasp, and which disallows grasp switches. If a dead end is encountered, we backtrack along the object manipulation plan until the previous grasp selection, and pick another grasp from the list.

IV. Experimental Evaluation

The planner architecture was implemented in MoveIt! [8]. The simulation was visualized in RViz and video-captures of solution paths recorded. The Interactive Markers interface of RViz was used to set problem specifications (ie., start and goal states).

We posed four problems to the robot which involved moving the ring around the card. One problem (TakeOut) has the ring hooked around one of the card holes, and we need to take it out from the card. The second problem requires taking out the ring, flipping it, and reinserting it to be hooked to the same hole as the start (FlipReinsert). The third problems require changing the card hole the ring is hooked around (ChangeHole). Finally, for the last problem, the ring starts hooked through both card holes; it must be taken out, flipped, and reinserted so it is hooked around both card holes (CenterHook).

Because the ring and card have thin features compared to the size of the workspace, we use a very fine resolution to check validity of motions. As a result, planning for one arm (bringing it to a grasp position, or setting it away) takes one second on average, with another half second for path simplification. Planning the motion for the ring alone (the first level of our approach) takes a half second on average.

We run each problem five times and measured average times and number of grasp changes needed to achieve the desired manipulation. These are given in table I. Some variability in solution paths exists because MoveIt has some tolerances in placing goal states, and also because the varying planning

times for grasp actions may change which grasps are considered successful. For all problems, the completion rate was complete; the planner always found a solution.

TABLE I
AVERAGES, STANDARD DEVIATION FOR PLAN TIMES; AVERAGES NUMBER OF GRASP SWITCHES

Problem	Avg time (s)	StD time (s)	Avg grasp count
TakeOut	85.6	16.2	4
FlipReinsert	91.2	26.9	7
ChangeHole	65.8	29.2	3
CenterHook	182	27.5	19

The largest amount of time is spent planning for arm grasps/ungrasps. When a grasp switch is necessary, we test (and therefore plan for) possible grasps with one arm, then follow the grasp to see which of the known and feasible grasps seems most promising. It was found that greedily choosing the grasp which allows the longest follow before needing a grasp switch is most often good enough, with few backtracks necessary. As a result, we estimate we can reduce planning times by separating grasp tests into a step of testing how long a grasp can follow the ring trajectory, and a step of planning to bring the arm to the grasp; then, delaying planning for the arm until it becomes necessary in the backtrack process will reduce the number of arm planning steps we do.

V. CONCLUSIONS

This work highlights some capabilities that may be interesting for a robotic manipulation software: the ability to store a roadmap for more complex tasks, as well as the possibility to augment the roadmap with some other information like grasp options; judging whether a given configuration can allow changing grasps, and finding such a configuration when necessary; a more efficient policy of keeping arms away from invalidating the solution path; allowing human assistance in selecting a repertoire of known configurations and maneuvers, useful for example in disassembly tasks.

Our approach trades probabilistic completeness for efficiency. Since we plan for the manipulated object first, there are no guarantees the robot will actually be able to move its arms in such a way so as to make the object follow the plan. In practice however, this was not an issue: the planning problems we posed were all solved. Future improvements of our method will include reducing the number of calls to the arm planning procedure.

In the class of problems considered here the solution often involves many maneuvers by the robot arms. A real world environment will likely not be completely static. Replanning and repairing plans becomes necessary, especially because the solution often requires narrow passages to be traversed and as such simple path deformation methods may be inadequate. Roadmaps on the other hand, if rich enough to capture a variety of possible behaviors, but small enough to be fast to queries, offer an efficient way to replan.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[3] I. A. Şucan, M. Kalakrishnan, and S. Chitta, "Combining planning techniques for manipulation using realtime perception," in *IEEE Intl. Conference on Robotics and Automation*, 2010, pp. 2895–2901.

[4] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE Intl. Conference on Robotics and Automation*, Taipei, Taiwan, September 2003.

[5] M. Saha and J. Latombe, "Finding narrow passages with probabilistic roadmaps: The small-step retraction method," in *Intl. Conference on Intelligent Robots and Systems*, Edmonton, Canada, August 2005, pp. 622– 627.

[6] L. Zhang, Y. Kim, and D. Manocha, "A hybrid approach for complete motion planning," in *Intl. Conference on Intelligent Robots and Systems*, San Diego, California, October 2007, pp. 7–14. [Online]. Available: http://www.ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4398944&arnumber=4399064&count=699&index=119

[7] J. N. Pires, *Industrial robots programming: building applications for the factories of the future*. Springer, 2007.

[8] Moveit. [Online]. Available: http://moveit.ros.org

[9] E. Plaku, M. Y. Vardi, and L. E. Kavraki, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Robotics: Science and Systems*, W. Burgard, O. Brock, and C. Stachniss, Eds. Atlanta, Georgia: MIT Press, June 2007, pp. 326–333.

[10] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park, "Tangent space RRT: A randomized planning algorithm on constraint manifolds," in *IEEE Intl. Conference on Robotics and Automation*, Shanghai, May 2011, pp. 4968–4973.

[11] D. Berenson, S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE Intl. Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 625–632.

[12] S. Dalibard and J.-P. Laumond, "Control of probabilistic diffusion in motion planning," in *Algorithmic Foundations of Robotics VIII*. Springer, STAR 57, 2009, pp. 467–481.

[13] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[14] S. Niekum, S. Chitta, A. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration," in *RSS*, 2013.

[15] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, pp. 343–352, 2009.

[16] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *IEEE Intl. Conference on Robotics and Automation*, Anchorage, Alaska, May 2010, pp. 5002–5008.

[17] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *IEEE Intl. Conference on Robotics and Automation*, Anchorage, Alaska, May 2010, pp. 2689–2696.

[18] T. S. Cambon, J.-P. Laumond, J. Corts, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7, pp. 729–746, 2004.

[19] F. Gravot, S. Cambon, and R. Alami, "asymov: A planner that deals with intricate symbolic and geometric problems," in *Intl. Symposium on Robotics Research*, 2003, pp. 100–110.

[20] J. Guitton and J.-L. Farges, "Taking into account geometric constraints for task-oriented motion planning," in *ICAPS Workshop on Bridging the gap Between Task And Motion Planning (BTAMP)*, 2009.

[21] K. Hauser and J.-C. Latombe, "Integrating task and prm motion planning: Dealing with many infeasible motion planning queries," in *ICAPS Workshop on Bridging the Gap Between Task and Motion Planning*, 2009.

[22] I. A. Şucan and L. E. Kavraki, "Mobile manipulation: Encoding motion planning options using task motion multigraphs," in *IEEE Intl. Conference on Robotics and Automation*, Shanghai, May 2011, pp. 5492–5498.

[23] L. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *IEEE Intl. Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[24] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics and Automation Magazine*, 2012. [Online]. Available: http://ompl.kavrakilab.org