Closed-Loop Global Motion Planning for Reactive Execution of Learned Tasks

Chris Bowen and Ron Alterovitz

***Abstract*— We present a motion planning approach for performing a learned task while avoiding obstacles and reacting to the movement of task-relevant objects. We employ a closed-loop sampling-based motion planner that acquires new sensor information, generates new collision-free plans that are based on a learned task model, and replans at an average rate of more than 10 times per second for a 7-DOF manipulator. The task model is learned from expert demonstrations prior to task execution and is represented as a hidden Markov model. During task execution, our motion planner quickly searches in the Cartesian product of the task model and a probabilistic roadmap for a plan with features most similar to the demonstrations given the locations of the task-relevant objects. We improve the replan rate by using a fast bidirectional search and by biasing the sampling distribution using information from the learned task model to construct high-quality roadmaps. We illustrate the efficacy of our approach by performing a simulated navigation task with a 2D point robot and a physical powder transfer task with the Baxter robot.**

I. Introduction

Robotic manipulators have the potential to assist people with a variety of routine tasks in both homes and workplaces. A key challenge for autonomously completing tasks in environments where people live and work is that objects in the environment are often not static—objects may move independently or be moved by people. A second key challenge is that many tasks must be performed in a manner that enforces constraints that humans are aware of from context and intuition; for example, the constraint that a glass of water must be kept approximately level when being carried in order to avoid spilling the water. A third key challenge is that homes and workspaces are often cluttered with obstacles, and these obstacles must be avoided to safely perform a task.

In this paper, we present a motion planning approach that addresses these challenges by using a learned task model and executing in a closed-loop manner, enabling the robot to be responsive to the movement of task-relevant objects in the environment while avoiding stationary obstacles. To learn a task model, our method requires a set of expert demonstrations, which can be performed in a static environment with no obstacles present. If the task-relevant objects are in different locations in different demonstrations, then the learner can generalize the task model to new locations. Once the task is learned, our motion planner can be executed in new environments with novel obstacles and can adjust to the movement of task-relevant objects as the task is being performed.

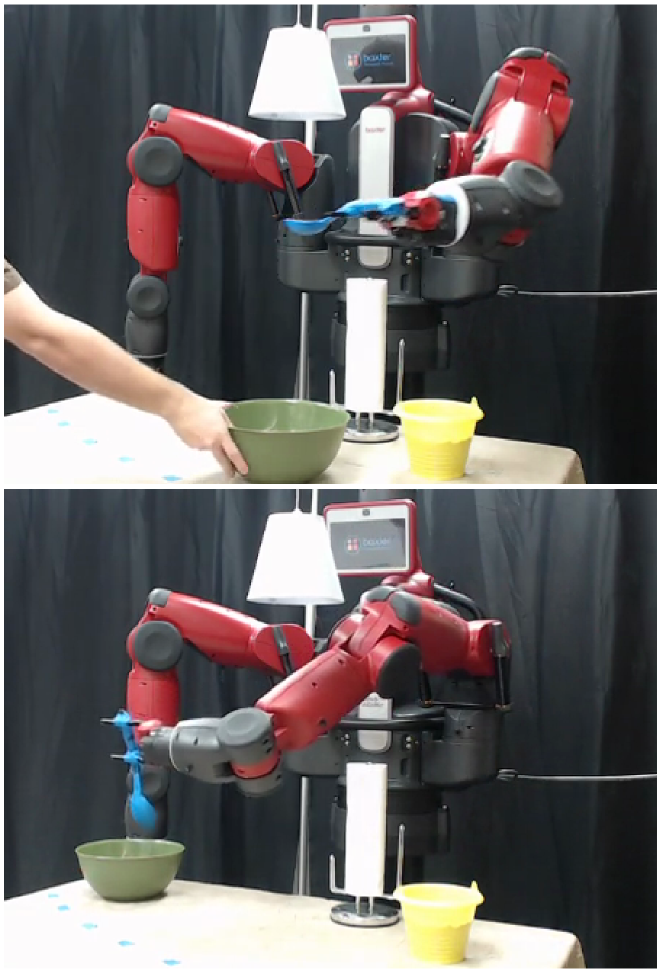

Fig. 1. As the Baxter robot performs the learned task of transferring powder from the yellow bucket to the green bowl using the blue spoon, a person moves the green bowl. Our method automatically replans in a closed-loop manner, enabling the robot to avoid obstacles and perform the learned task even when task-relevant objects are moved mid-task.

Based on prior work [1], we consider a task model consisting of a sequence of probability distributions over the configuration of the robot given the state of the environment. Specifically, we consider features defined by both the configuration of the robot and the position of the end-effector relative to task-relevant objects (e.g., the bowl in the powder transfer task in Fig. 1). We learn not only the average values of these features over time, but their variances, under the assumption that more consistent features in the demonstrations should similarly be reproduced consistently

This research was supported in part by the National Science Foundation (NSF) under awards IIS-1117127 and IIS-1149965.

Chris Bowen and Ron Alterovitz are with the Department of Computer Science, The University of North Carolina at Chapel Hill, NC, USA {cbbowen,ron}@cs.unc.edu

during execution. For instance, in the powder transfer task in Fig. 1, the orientation of the spoon is likely to have been consistently level in the demonstrations during the time that the expert was transferring powder.

During execution, our motion planner uses these distributions to define a cost function which is minimized by an interactive-rate sampling-based planner. The planner finds solutions that approach global optimality, in contrast to approaches that only guarantee local optimality and may become caught in the basin of attraction of a bad solution. We search for plans in a roadmap defined by the Cartesian product of the learned task model, represented as a hidden Markov model, and a probabilistic roadmap over the robot's configuration space. To consider the movement of task-relevant objects, we continuously replan by updating the roadmap edges and costs and then searching for a new plan. This replanning approach computes plans that avoid stationary obstacles while explicitly considering the motion of task-relevant objects.

The robot replans in real-time, averaging more than 10 plans per second in our experiments, by leveraging information in the task model and using appropriate data structures and algorithms. First, we employ a sampling distribution biased toward low-cost regions of configuration space to produce a small, high-quality roadmap. Second, we use a parallel bidirectional search over an implicit graph combined with lazy computation to quickly and globally search for optimal plans.

Our approach differs from prior work by incorporating learning from demonstrations, obstacle avoidance, and global sampling-based replanning in a unified method. We apply our method to both a simulated scenario and to the Baxter robot.

II. Related Work

Sampling-based methods have been highly successful for computing feasible (and optimal) motion plans for a wide variety of robots, including manipulators with many degrees of freedom [2], [3], [4]. While most sampling-based motion planners that consider optimality aim to minimize metrics such as Euclidean distance in the workspace or configuration space, some methods have investigated incorporating more general task-based cost functions. Several approaches are based on rapidly exploring random trees (RRTs) [2], a highly successful sampling-based method for computing feasible, obstacle-avoiding trajectories. Transition-based RRT (T-RRT) [5] biases expansion of an RRT to low cost regions of the configuration space cost map, and Mainprice et al. used T-RRT to generate natural motions based on a predefined cost map for human robot interaction [6]. RRTs have also been used in conjunction with analytically-defined task constraints [7] and with symbolic representations of manipulation strategies [8]. Recent sampling-based motion planners have also investigated integrating motion constraints and properties learned from demonstrations. Claasens extended RRT to sample only inside a user-specified number of standard deviations of a mean demonstrated trajectory [9], Berenson et al. integrated local optimization with an RRT to find low-cost paths over cost maps [10], Scholz et al. incorporated gradient descent into an RRT to locally optimize a specified objective function [11], and Şucan and Chita investigated sampling strategies that enforce constraints [12]. Finally, RRTs have been used for replanning when the environment changes [13], [14], [15]. However, plans produced by RRT methods are almost surely suboptimal, even as computation time approaches infinity.

We employ an asymptotically optimal sampling-based motion planner, meaning the computed plan is guaranteed to approach a globally optimal plan (based on the given cost metric) as computation time is allowed to increase. Karaman and Frazzoli proposed asymptotically optimal motion planning algorithms such as RRG and PRM* that guarantee asymptotic optimality [4]. Asymptotically optimal motion planners avoid the suboptimal plans resulting from local minima that can occur when using potential field methods [3] or resulting from sampling-based planners not designed for asymptotic optimality like RRT [4]. Related work has investigated asymptotically optimal planners that balance exploration and refinement [16], asymptotic near optimal planners using smaller roadmaps [17], the near-optimality of solutions in finite time [18], and anytime solution optimization [19].

Our method integrates a learned hidden Markov model (HMM) representing a task with a sampling-based motion planner to guarantee asymptotic optimality. HMMs have previously been used for motion generation (e.g. [20], [21]), but prior approaches, unlike our proposed method, do not simultaneously guarantee global optimality while enabling fast replanning. A number of methods learn high level tasks from demonstrations (e.g., [22]). For execution, these methods may use visual servoing in conjunction with subtask-specific motion planners [23]. Our method differs in that we learn the motions of these subtasks rather than learning a high level task and explicitly specifying subtasks. Dynamical systems have been used to learn motions from demonstrations, e.g. [24], [25], and [26]. Such systems have also been extended to avoid obstacles in [27], but this approach retains no notion of global optimality.

III. Method Overview

A. Problem Statement

Let $\mathcal{Q} \subseteq \mathbb{R}^d$ be the d-dimensional configuration space of the robot. Let $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ denote the subspace of the configuration space for which the robot is not in collision with an obstacle in the current execution environment. Let $\mathbf{q} \in \mathcal{Q}$ denote a configuration of the robot. We assume the robot is capable of sensing the positions of L task-relevant objects, called *landmarks* (such as the green bowl in Fig. 1). We also assume the robot is holonomic with position-controlled joints and that obstacles in the environment (in contrast to task-relevant objects) do not move.

During execution (described in Sec. IV), our objective is to compute a trajectory Φ in the robot's configuration space from a start configuration $\mathbf{q}_{\text{start}} \in \mathcal{Q}_{\text{free}}$ to a goal configuration $\mathbf{q}_{\text{goal}} \in \mathcal{Q}_{\text{free}}$ such that the trajectory (1)

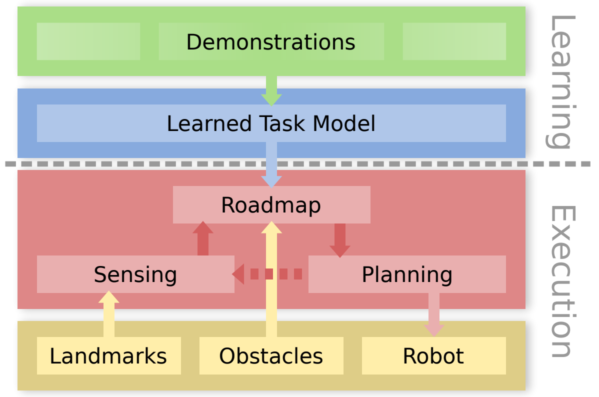

Fig. 2. General overview of the components of the system, with expert input shown in green, task learning shown in blue, execution shown in red, and environment shown in yellow. Note that the task model does not depend on the environment and so need only be learned once per task. It can then be used for execution across many environments. All arrows indicate execution flow and solid arrows indicate data flow.

avoids obstacles in the robot's current environment and (2) moves relative to the task-relevant objects in a manner that successfully accomplishes the task.

To address this challenge, our approach consists of two major phases: learning a task model from a set of demonstrations followed by task performance in the execution environment. Note that the task model is not tied to a specific environment, so it only needs to be learned once per task. It can then be used for execution across many environments. Fig. 2 illustrates an overview of the approach.

During the learning phase, the user provides demonstrations of a task class. These demonstrations can be performed with no obstacles present, and the environment in each demonstration can be static. As introduced in [1] and summarized in Sec. III-B, we use statistical methods to learn a task model, which can be mapped to new environments with previously unseen obstacles and in which task-relevant objects may be in different locations.

In the execution phase, presented in Sec. IV, the robot first senses its environment to collect sufficient information to perform collision detection and to compute costs in the current environment based on the learned task model. The robot then computes a collision-free motion plan based on the learned costs. We then enter the loop shown in red in Fig. 2; the robot executes the current plan while, in a closed-loop manner, replanning based on the learned costs as task-relevant objects might move in the environment.

B. Learned Task Model

To learn a task, we consider a reformulation of the method originally presented in [1], wherein a task was learned from a set of demonstrations of a human kinesthetically guiding the robot through the task. For each demonstration, we record a sequence of configurations evenly-spaced in time as well as an annotation $\mathbf{a}$ that encodes the positions of task-relevant objects during the demonstration. We then time-align the demonstrations and use them to estimate a sequence of multivariate Gaussian distributions $\mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ in a motion feature space consisting of the robot's configuration and end-effector position relative to task-relevant objects in the environment.

$$\begin{tikzcd} s_1 \arrow[loop above, "p_{1,1}"] \arrow[r, "p_{1,2}"] \arrow[d, squiggly] & s_2 \arrow[loop above, "p_{2,2}"] \arrow[r, "p_{2,3}"] \arrow[d, squiggly] & s_3 \arrow[loop above, "p_{3,3}"] \arrow[d, squiggly] \\ \mathcal{N}(\mu_1, \mathbf{\Sigma}_1) & \mathcal{N}(\mu_2, \mathbf{\Sigma}_2) & \mathcal{N}(\mu_3, \mathbf{\Sigma}_3) \end{tikzcd}$$

Fig. 3. Hidden Markov model with 3 states and Gaussian distributed observations.

In this paper, for ease of integration with a constantly updating motion planning roadmap, we reformulate this approach as learning a hidden Markov model (HMM) with a restricted structure (see Fig. 3). We assume the model has S sequential states wherein each state s_i has nonzero transition probabilities only to itself and the next state s_{i+1}. This induces a linear order structure to the model corresponding to time. As such, we refer to these states as *time steps*. For our experiments, we let the observed outputs in each state s_i be distributed according to a multivariate Gaussian distribution $\mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ in motion feature space.

This learned model will be used to compute a cost function for motion planning during task execution. The cost function is parameterized by the annotation $\mathbf{a}$, which will vary across execution environments as well as vary during execution when a task-relevant object moves. The cost function [1] is defined such that a trajectory which minimizes it, maximizes the likelihood in the learned model, and thereby should successfully perform the task in the execution environment with high probability.

IV. Closed-Loop Replanning with Learned Costs

For motion planning, our method first builds a spatiotemporal roadmap in which the edge costs are set based on the learned task model. During task execution, we efficiently update the roadmap edge costs and perform searches on the roadmap in a closed-loop manner to account for the movement of task-relevant objects. We now describe each of the components of the closed-loop motion planning approach.

A. Spatiotemporal Roadmap

We employ a variation of a probabilistic roadmap (PRM) [28], which we chose to adapt because it produces globally optimal plans to within the roadmap resolution. A roadmap is a graph in which vertices represent the states of the robot and edges represent feasible local plans between these states. In the simplest case these local plans are just straight line trajectories in configuration space.

A traditional roadmap is undirected and is constructed as follows. First, N configurations $\{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_N\}$ are randomly sampled from $\mathcal{Q}_{\text{free}}$ via rejection sampling. Then, edges are constructed between all configurations $\mathbf{q}_i$ and $\mathbf{q}_j$ for which $\|\mathbf{q}_i - \mathbf{q}_j\|_D < \epsilon$ if a feasible local plan can be

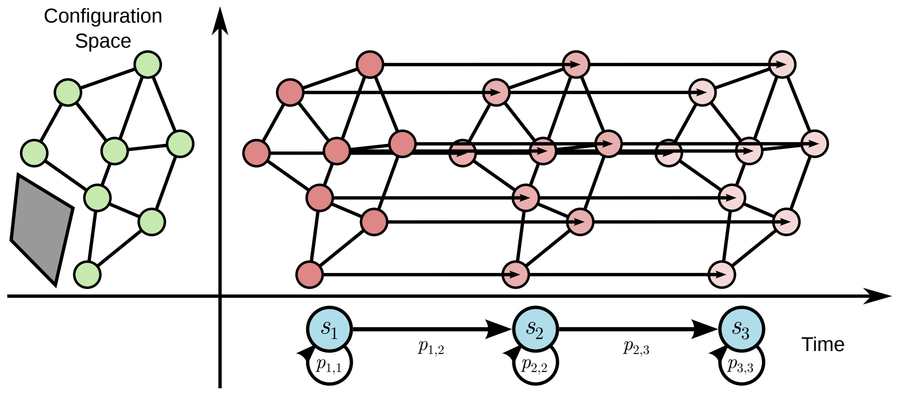

Fig. 4. Cartesian product of the configuration space roadmap and the graph of the learned task model.

found. We construct exactly such a roadmap, which we will call the *spatial roadmap*.

To accommodate dependence on the time step in the learned task, we also define a *temporal roadmap*. This is not a roadmap in the traditional sense, but rather the graph representation of the Markov chain implied by the task model, where time steps correspond to vertices in the graph. Because the transitions in the Markov chain are directed, the temporal roadmap is a directed graph.

Finally, we define a *spatiotemporal roadmap*, a directed graph that combines the information in the spatial and temporal roadmaps. The vertices of the spatiotemporal roadmap are each defined by a pair composed of a vertex from the spatial roadmap and a vertex from the temporal roadmap. The set of edges are given by the vertex-wise union of edges in the spatial and temporal roadmaps (see Fig. 4). Put another way, the spatiotemporal roadmap is the Cartesian product of the spatial and temporal roadmaps. Such a roadmap is necessary because the state of the robot needs to incorporate the task progress (see Sec. V-A).

This full roadmap can be quite large, but because of its regular structure, it need not be explicitly constructed. Rather, we can implicitly traverse it by keeping track of vertices in the constituent roadmaps. This approach has multiple advantages over directly sampling in the product of configuration space and time. First, because the full roadmap does not need to be explicitly constructed, it uses less memory and exhibits better locality of reference. Second, because every vertex and edge in the spatial roadmap is effectively duplicated across all time steps, fewer collision queries are required. Finally, small additions to either constituent roadmap are immediately reflected as larger additions to the full roadmap without the need to perform an explicit construction, making updates fast. As an aside, we also experimented with the tensor product of the roadmaps, which while more natural, produced many more edges and adversely impacted search performance.

B. Cost Function

In a traditional roadmap, the goal is to find shortest paths, so edges in the roadmap are assigned costs based on the lengths of the local plans they represent. We wish to find paths which are most likely to successfully perform the task, so choose edge costs based on the learned task model. Specifically, we define a notion of cost which, when minimized, maximizes probability of successfully executing the task as defined by the learned model.

By the construction of the Cartesian product, each edge is from a configuration $\mathbf{q}$ and time step s to another configuration $\mathbf{q}'$ and time step s'. Our definition of edge cost will be based on the negative log probability of entering state s' and observing configuration $\mathbf{q}'$ after being in state s and observing configuration $\mathbf{q}$, which simplifies as follows:

$$\begin{aligned} &-\log(\mathrm{P}(\mathbf{q}', s' \mid \mathbf{q}, s)) = \\ &-\log(\mathrm{P}(\mathbf{q}' \mid s') \cdot \mathrm{P}(s' \mid s)) = \quad \text{(Markov Property)} \\ &-\log \mathrm{P}(\mathbf{q}' \mid s') - \log \mathrm{P}(s' \mid s). \end{aligned}$$

We consider the log probability because the shortest path algorithm will minimize the *sum* of the edge costs while joint probabilities (under independence assumptions) are multiplicative, as seen above. We then negate the result to formulate the problem as a minimization with non-negative edge costs. However, recall that edges in the spatial roadmap represent local plans. So for such edges, we use the line integral of the negative log probability along the local plan.

C. Real-Time Execution

During execution, we update the roadmap using the latest sensed information and query the roadmap in a closed-loop manner. Hence, the method updates with high frequency the path being executed. In Fig. 5 we show a schematic of the computation during real-time execution.

Updates to the roadmap take two forms, *expansion* and *re-evaluation*.

Expansion occurs both initially and in each replanning cycle to add waypoints to the spatial roadmap. During expansion we first numerically compute a locally optimal trajectory, analogous to the guiding path in [1], based on the sensed locations of the task-relevant objects. This locally optimal trajectory may intersect obstacles. We then sample a fixed number of new configurations from a mixture of Gaussian distributions each with mean on this locally optimal trajectory and covariances estimated from the demonstrations. These new configurations are checked for collision

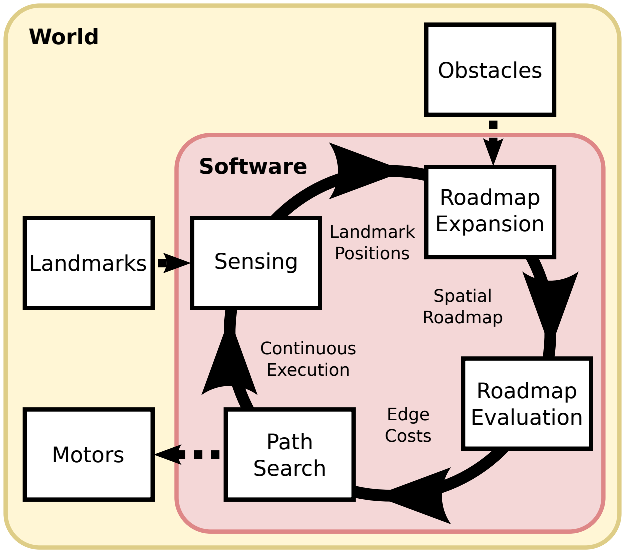

Fig. 5. Flow of computation during real-time execution.

with obstacles and are added to the spatial roadmap as in a traditional PRM. This biased sampling allows the planner to produce high-quality plans without expending much computational effort in regions of configuration space which the robot is highly unlikely to traverse. Expansion after initial roadmap construction is important because the sampling density used initially may not be sufficient to produce high-quality plans, particularly when the environment changes dramatically.

Re-evaluation occurs in each replanning cycle to update roadmap edge costs and compute an updated path. At the beginning of each replanning cycle, the positions of the task-relevant objects are obtained from the sensor. Because the edge costs depend on these positions, the edge costs must be recomputed when the task-relevant objects move. We recompute the edge costs lazily while searching for the shortest path in the roadmap from the current configuration to the goal. We compute the shortest path using a bidirectional search based on Dijkstra's algorithm. Together, the lazy cost evaluation and bidirectional search greatly reduce the portion of the roadmap which must be explored and the number of edge costs which must be computed. Furthermore, this search (and thus cost evaluation) is parallelized across two cores, cutting computation time nearly in half.

Unlike many methods for real-time planning or control, our approach is globally optimal in that it selects the true minimal cost path present in the roadmap. Consequently, our method may consider multiple homotopic classes of paths and can avoid being trapped in a local minimum. Because we use a PRM, our approach is also asymptotically optimal in the sense that the plans produced approach optimality as the size of the roadmap used increases. We do not use the shrinking edge connection radius of PRM* [4] because the roadmap only grows slowly after initial construction, so a good fixed connection distance can be determined offline, prior to execution. While we only build and expand the roadmap for short periods of time, asymptotic optimality of the planner allows the method to find better solutions,

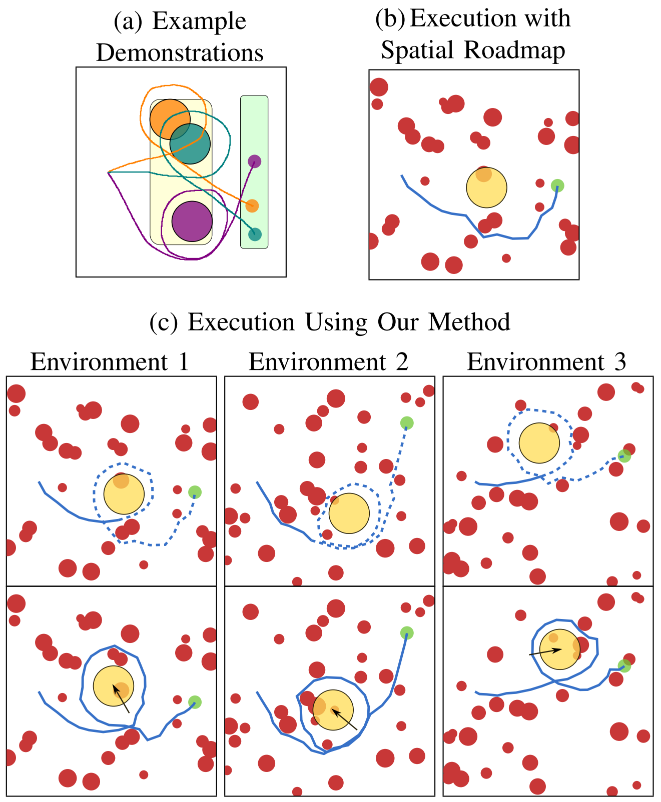

Fig. 6. (a) Three of the 7 demonstrations for the simulated navigation task as well as the beacon and goal sampling regions. The demonstrated paths start at a fixed location, move counter-clockwise around a specified beacon, and then move to a specified goal. (b) Execution without the temporal roadmap and using a temporal alignment heuristic [1] with obstacles (red), a beacon (yellow), and a goal (green). (c) Three executed trajectories (solid blue) and planned trajectories (dashed blue) as computed by our method, before (top) and after (bottom) the beacon was moved.

approaching the global optimum, as computational power increases.

Our execution is real-time in the sense that we impose a firm deadline on the planner, from sensing to execution, to ensure the robot is never acting on sensor information that is excessively out of date. A missed deadline manifests as a pause while the robot waits for a new plan to be sent to the motors for actuation. If the replanning cycle finishes before the deadline, then the next replanning cycle begins immediately. In our implementation, we used a deadline of 250 milliseconds, although in our experiments we were on average able to achieve replanning cycles substantially faster than the deadline.

V. Results

To show the applicability of the method, we ran it on a simulated 2D navigation task and on a physical task using the Baxter robot [29]. All computation was performed using a C++ implementation on a PC with two 2.0 GHz Intel Xeon E5-2620 processors.

A. Simulated Navigation Task

We consider a point robot that is to navigate on a 2D plane by starting at a fixed location, moving counter-clockwise completely around a beacon without intersecting it, and

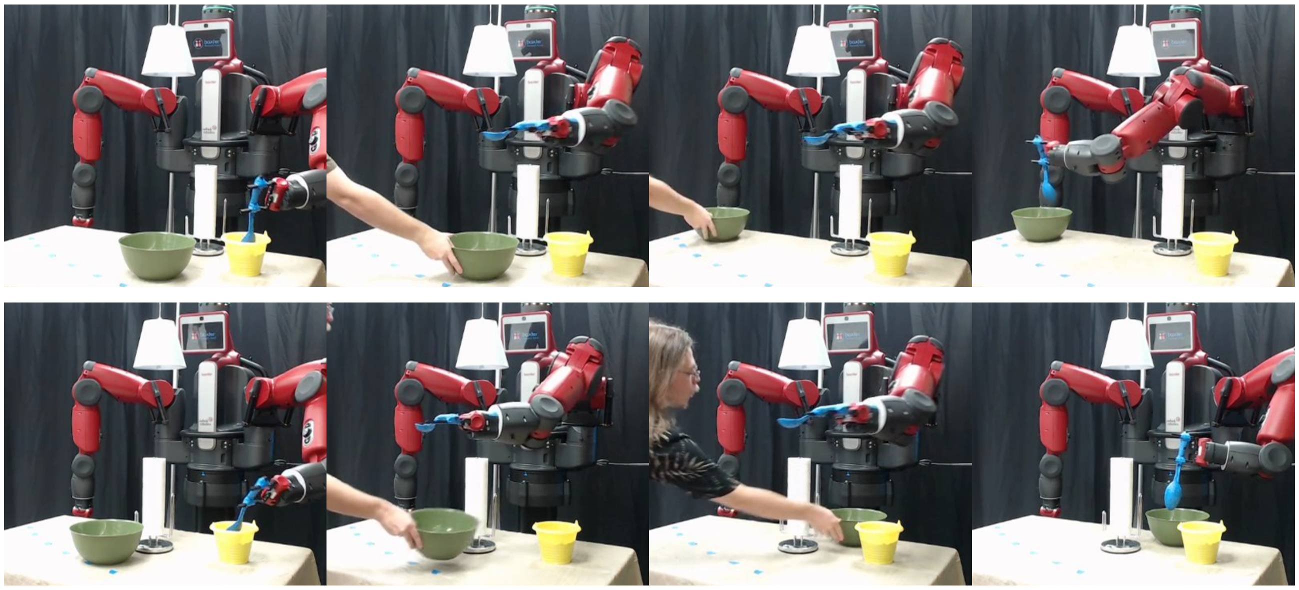
Fig. 7. Two example executions for the powder transfer task, from a bucket (yellow) to a bowl (green).

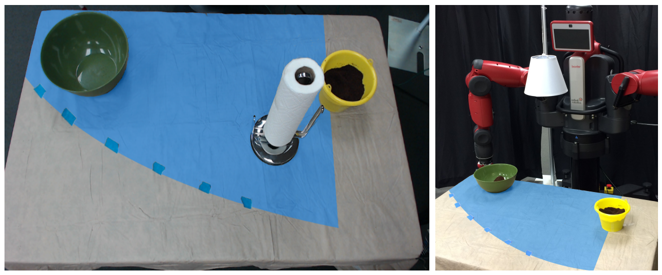
Fig. 8. Environment for transfer task with the permissible region for both the green bowl and paper towel roll highlighted in blue.

	Success Rate	Avg. Pauses	Avg. Update
Full Method	80%	0	89 ms
No Bidirectional Search	50%	16	174 ms
No Obstacle Avoidance	10%	0	85 ms
No Biased Sampling	0%	–	–
No Replanning	0%	–	–

Fig. 9. Success rates, average pauses, and average update periods for variants of the method. The additional failures without bidirectional search were due to insufficient reaction time when the bowl was moved, and without biased sampling, the method was unable to find successful plans even after 5 minutes of roadmap computation time.

then stopping at a specified goal. In the learning phase, the feature space consisted of the absolute position of the robot and its position relative to two landmarks, corresponding to the beacon and intended goal position specified by the annotations. For the demonstrations, we manually performed 7 successful trajectories with beacon locations randomly sampled from the yellow region and goal locations randomly sampled from the green region in Fig. 6(a).

In the execution phase, the test environment included 32 circular obstacles which were not present in the demonstrations. We randomly generated 15 test cases with different obstacle locations and with randomly chosen beacon and goal locations (sampled independently from the locations chosen for demonstrations but using the same regions). A quarter of the way through the trajectory (approximately when the robot begins encircling the beacon) we moved the beacon 10% of the width of the environment in a random direction.

The method successfully accomplished the task in all 15 test cases. Several representative executions are shown in Fig. 6(c). Due to the density of the obstacles, in most of the test cases it was necessary for the method to change homotopic classes when replanning, illustrating the need for global planning rather than local refinement of plans.

This task also illustrates the need to incorporate task progress into the robot state. Planning in configuration space alone is not sufficient for correct execution because a successful path necessarily crosses itself. Using the temporal alignment heuristic from [1] skips an important portion of the task as shown in Fig. 6(b).

B. Powder Transfer Task

In the second task, the goal was to scoop powder from a small bucket and transfer it into a bowl using a spoon. To successfully perform the task, the robot was required to transfer the powder without spilling while avoiding obstacles in the environment. The feature space consisted of the configuration of the robot and the position of the spoon tip relative to the bowl, for a total feature dimension of 10. We provided the method with 11 successful kinesthetic demonstrations from which to learn.

At the beginning of execution the green bowl was randomly placed on the reachable surface of the table and tracked using a Kinect sensor. For obstacles, we affixed a hanging lamp shade above the table and randomly placed a vertical roll of paper towels on the table (see Fig. 8). If a random obstacle location intersected the bowl, we rejected the location and sampled a new location. During task execution, when the robot positioned the spoon above the bowl but before it began dumping the contents, we quickly moved the bowl to a different random location.

In 8 out of the 10 tests, the method was successful. In the 2 failed tests, the method was unable to find a feasible trajectory to the bowl due to a narrow passage created by the obstacles, and so executed trajectories that failed to properly transfer the powder. We show the average update period, defined as the total time required to perform sensing and replan, for different variants of the method along with success rate and pauses due to missed deadlines in Fig. 9.

This task, as well as another task, are shown in the video associated with this paper.

VI. Conclusion

We presented a closed-loop motion planning approach that is applicable to execution of learned tasks in which task-relevant objects move during execution. We illustrated the efficacy of the approach for a simulated navigation task and for a physical powder transfer task with the Baxter robot.

In future work, we plan to consider the problem of extrapolating the position of the task-relevant objects and incorporating the noise in these predictions to produce better plans and to accommodate occlusion. We will also investigate methods to make replanning faster, including using bidirectional A* with a task-specific admissible heuristic. The additional performance could enable us to effectively handle faster moving objects. It could also enable us to extend the method to handle not only moving task-relevant objects but also moving obstacles. We also plan to consider generalizations of the learned task model to consider more complex tasks that require integration of task and motion planning.

ACKNOWLEDGMENT

The authors thank Jenny Womack, Sachin Patil, Pierre Berthet-Rayne, Wen Sun, and Christophe Cordero for their input.

References

[1] G. Ye and R. Alterovitz, “Demonstration-guided motion planning,” in *Int. Symp. Robotics Research (ISRR)*, Aug. 2011.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

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

[4] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[5] L. Jaillet, J. Cortes, and T. Simeon, “Sampling-based path planning on configuration-space costmaps,” *IEEE Trans. Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

[6] J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon, “Planning human-aware motions using a sampling-based costmap planner,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2011, pp. 5012–5017.

[7] M. Stilman, “Global manipulation planning in robot joint space with task constraints,” *IEEE Trans. Robotics*, vol. 26, no. 3, pp. 576–584, Jun. 2010.

[8] R. Jakel, S. R. Schmidt-Rohr, M. Losch, and R. Dillmann, “Representation and constrained planning of manipulation strategies in the context of programming by demonstration,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2010, pp. 162–169.

[9] J. Claassens, “An RRT-based path planner for use in trajectory imitation,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2010, pp. 3090–3095.

[10] D. Berenson, T. Simeon, and S. S. Srinivasa, “Addressing cost-space chasms in manipulation planning,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2011, pp. 4561–4568.

[11] J. Scholz and M. Stilman, “Combining motion planning and optimization for flexible robot manipulation,” in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2010, pp. 80–85.

[12] I. A. Şucan and S. Chitta, “Motion planning with constraints using configuration space approximations,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 1904–1910.

[13] D. Ferguson, N. Kalra, and A. Stentz, “Replanning with RRTs,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2006, pp. 1243–1248.

[14] M. Zucker, J. Kuffner, and M. Branicky, “Multipartite RRTs for rapid replanning in dynamic environments,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 2007, pp. 1603–1609.

[15] K. Hauser, “On responsiveness, safety, and completeness in real-time motion planning,” *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, Sep. 2011.

[16] R. Alterovitz, S. Patil, and A. Derbakova, “Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2011, pp. 3706–3712.

[17] J. D. Marble and K. E. Bekris, “Asymptotically near optimal planning with probabilistic roadmap spanners,” *IEEE Trans. Robotics*, vol. 29, no. 2, pp. 432–444, Apr. 2013.

[18] A. Dobson and K. E. Bekris, “A study on the finite-time near-optimality properties of sampling-based motion planners,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 1236 – 1241.

[19] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki, “Anytime solution optimization for sampling-based motion planning,” in *IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2013, pp. 5068 – 5074.

[20] S. Calinon, F. D’halluin, D. G. Caldwell, and A. G. Billard, “Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework,” in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2009, pp. 582–588.

[21] D. Kulic, W. Takano, and Y. Nakamura, “Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains,” *Int. J. Robotics Research*, vol. 27, no. 7, pp. 761–784, Jul. 2008.

[22] S. Benson, “Inductive learning of reactive action models,” in *Proc. Int. Conf. Machine Learning (ICML)*, 1995, pp. 47–54.

[23] S. Ekvall and D. Kragic, “Robot learning from demonstration: A task-level planning approach,” *Int. J. Advanced Robotic Systems*, vol. 5, no. 7, pp. 223–234, 2008.

[24] M. Hersch, F. Guenter, S. Calinon, and A. Billard, “Dynamical system modulation for robot learning via kinesthetic demonstrations,” *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1463–1467, Dec. 2008.

[25] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, “Online movement adaptation based on previous sensor experiences,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sep. 2011, pp. 365–371.

[26] A. Shukla and A. Billard, “Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 424–440, Mar. 2012.

[27] S. M. Khansari-Zadeh and A. Billard, “A dynamical system approach to realtime obstacle avoidance,” *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, May 2012.

[28] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high dimensional configuration spaces,” *IEEE Trans. Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[29] Rethink Robotics, “Baxter Research Robot,” http://www.rethinkrobotics.com/products/baxter-research-robot/, 2013.