Demonstration-Guided Motion Planning

Gu Ye and Ron Alterovitz

Abstract We present demonstration-guided motion planning (DGMP), a new framework for planning motions for personal robots to perform household tasks. DGMP combines the strengths of sampling-based motion planning and robot learning from demonstrations to generate plans that (1) avoid novel obstacles in cluttered environments, and (2) learn and maintain critical aspects of the motion required to successfully accomplish a task. Sampling-based motion planning methods are highly effective at computing paths from start to goal configurations that avoid obstacles, but task constraints (e.g. a glass of water must be held upright to avoid a spill) must be explicitly enumerated and programmed. Instead, we use a set of expert demonstrations and automatically extract time-dependent task constraints by learning low variance aspects of the demonstrations, which are correlated with the task constraints. We then introduce multi-component rapidly-exploring roadmaps (MC-RRM), a sampling-based method that incrementally computes a motion plan that avoids obstacles and optimizes a learned cost metric. We demonstrate the effectiveness of DGMP using the Aldebaran Nao robot performing household tasks in a cluttered environment, including moving a spoon full of sugar from a bowl to a cup and cleaning the surface of a table.

1 Introduction

Over 10 million people need assistance with activities of daily living (ADLs) such as cooking, cleaning, and dressing in the United States. And this number is expected to grow dramatically worldwide as the number of elderly individuals continues to increase rapidly. Providing assistance to these individuals for ADLs is currently very labor intensive and often requires moving an individual from their home to an

Gu Ye and Ron Alterovitz
Department of Computer Science, The University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599, USA. e-mail: {guye,ron}@cs.unc.edu

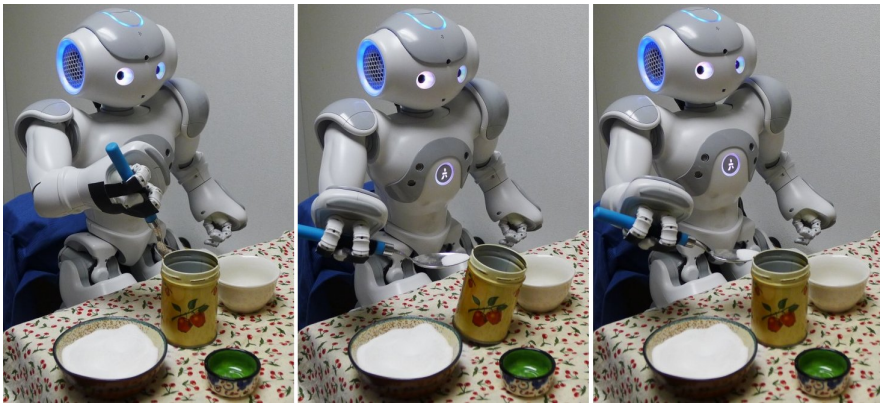

Fig. 1 We consider the household task of using a spoon to transfer sugar from a bowl to a cup on a table. Using standard motion planning algorithms without explicitly programming task constraints will fail since the spoon will not be kept level, causing the sugar to be spilled on the table (left). Methods based on learning from demonstrations are often unable to compute collision-free trajectories when novel obstacles are introduced in new locations not considered in the demonstrations (middle). Our method effectively combines motion planning with learning from demonstrations to avoid obstacles and keep the spoon level to successfully transfer the sugar (right).

impersonal institution, which comes at a significant cost to society. Recently developed personal robots have the potential to assist individuals with a variety of ADLs. However, the algorithms to control these robots for autonomous, safe assistance with household tasks is still a work in progress in the robotics research community.

A key challenge is the development of algorithms that enable personal robots to plan motions in unstructured environments to accomplish tasks currently performed by humans. These tasks often involve significant constraints on motion that humans are aware of from context and intuition. For example, when carrying a plate of food from the kitchen to the dining room, a person knows that tilting the plate sideways, while feasible, is undesirable. The robot must be aware of such task constraints and at the same time avoid unforeseen obstacles in unstructured environments.

We present a new approach that unifies ideas from two fields: robot motion planning and robot learning from demonstrations. The motion planning community has developed highly successful sampling-based methods that efficiently compute feasible plans that avoid obstacles. However, motion planning methods typically do not consider any task constraints unless explicitly programmed. As shown in Fig. 1, direct application of standard motion planning algorithms to a robot transferring a spoonful of sugar from a bowl to a cup will result in the robot spilling the sugar unless the robot is explicitly programmed to keep the spoon level. While such task constraints can be innocuous to program individually, requiring expert programmers to anticipate and program all such constraints for robots operating in human environments would be prohibitive. In contrast, the learning from demonstration community has developed highly effective approaches for extracting from multiple demonstrations the key motions required to accomplish a task. Such methods can

explicitly or implicitly learn task constraints, such as the fact that spoons full of sugar should be kept level to avoid spillage. However, methods based on learning from demonstrations often falter when the robot must accomplish the task in a new, cluttered environment where unforeseen obstacles compel the robot to move outside the range of motions included in the input demonstrations. Our new approach, *Demonstration-Guided Motion Planning* (DGMP), combines the strengths of methods in motion planning and in learning from demonstration to both (1) avoid novel obstacles in cluttered environments, and (2) learn and maintain critical aspects of the motion required to successfully accomplish a task.

To teach the robot the motions necessary to assist in a household task, we use kinesthetic demonstrations; the robot's joints are placed in a passive mode and the demonstrator moves the robot's arms and torso through each step of the task. Kinesthetic demonstrations are ideally suited for teaching household assistance tasks. They do not require the human expert (e.g. an occupational therapist) to learn complicated computer/robot programming. Instead, they rely only on physical, real-world demonstrations that are an intuitive and user-friendly way for human experts to teach robots to perform new tasks.

The emphasis of DGMP is not on learning high-level task decompositions or low-level controls from the demonstrations, but rather to learn the constraints on the robot's motion that are required to accomplish a repeated task. From a set of demonstrations, we apply statistical analysis to learn low and high variance components of the motion. Low variance regions correspond to implicit constraints that should be satisfied when the motion plan is executed. We use these variances to define a cost metric over the configuration space of the robot.

After the learning phase, DGMP uses a new motion planning algorithm, a multi-component rapidly-exploring roadmap (MC-RRM). The planner guarantees obstacle avoidance and incrementally improves the plan, guaranteeing convergence to the optimal feasible solution for the cost metric as computation time is allowed to increase.

We demonstrate the effectiveness of DGMP using the Aldebaran Nao robot performing household tasks in a cluttered environment, including transferring sugar from a bowl to a cup using a spoon and wiping a table. Our results demonstrate that unlike using motion planning or learning from demonstration methods in isolation, our unified approach converges to an optimal solution and offers a significantly higher success rate in accomplishing tasks that involve both learning of task constraints as well as obstacle avoidance.

2 Related Work

Our framework bridges robot motion planning with robot learning from demonstrations. Learning from demonstration methods have been highly successful in enabling robots to learn task constraints and imitate task motions [6, 4]. Motion planning methods have been effective at computing feasible motions from a start configuration to a goal configuration while avoiding obstacles [10, 16].

Demonstrations can provide examples of the motion required to accomplish a task, and these demonstrations can be used to computationally learn a control policy that will enable a robot to autonomously execute the task motion subject to real-world noise and disturbances. Inverse reinforcement learning has been used to estimate the unknown objective function of a control policy from demonstrations in environments with complex dynamics. This approach, sometimes called apprenticeship learning, has been applied to learn control policies for car driving [2], helicopter acrobatics [12], and robotic knot tying [24]. Another approach models the variations across demonstrated motion trajectories using a Gaussian Mixture Model (GMM) and then uses Gaussian Mixture Regression (GMR) to estimate the ideal trajectory and a corresponding controller [7]. Our approach builds on the GMM/GMR workflow from Calinon et al. for extracting local trajectories expressed in coordinate systems relative to objects in the environment [7]. The GMM/GMR approach has been applied to manipulation tasks such as moving chess pieces or feeding a doll and is robust to movement of obstacles included in the demonstrations [8]. However, these methods lack the ability to avoid novel obstacles that were not explicitly considered during the demonstrations, which is critical for motion planning in household environments.

Recent methods have used learning from demonstration methods to consider previously unseen obstacles, but existing methods are limited either to low dimension spaces, place limitations on the locations of obstacles, or do not allow for time-dependent task-space constraints. Prior work has investigated using global search methods such as A* or D* where path costs are learned from demonstrations. This approach has been successfully applied to navigating cars in a parking lot [1], maneuvering off-road vehicles in complex terrain [23], and generating natural motions for animated characters [17]. However, these methods do not consider time-dependent constraints and they discretize the state space, which does not scale well to higher degree of freedom systems like some personal robots. Another approach models demonstration trajectories as fluid currents in the task space and performs fluid dynamic simulation to derive a control policy [19]. Fluid simulation considers obstacles, but performance is unclear if obstacles pass through the demonstration trajectories.

In contrast to robot learning methods that have focused on extracting meaningful data from demonstrations, motion planning focuses on computing feasible plans that avoid obstacles. In particular, sampling based methods have been very successful for a wide variety of problems involving high-DOF robots [16, 10]. Prior work has investigated the use of RRTs combined with learned metrics to generate paths, but these methods are guaranteed to converge to a suboptimal solution [15]. One approach extends RRTs to sample only inside a provided number of standard deviations of a mean demonstrated trajectory, but may not find a feasible solution even if one exists [11]. RRTs have also been used in conjunction with task-based symbolic constraints [14]. Transition-based RRT (T-RRT) [13] is a sampling-based motion planner over cost maps that biases expansion of the tree to low cost regions of the configuration space. T-RRT can be used to generate natural motions by using a predefined human robot interaction cost metric [18]. GradienT-RRT extends T-RRT to

use the gradients of the cost function to locally optimize the trajectory and facilitate finding solutions through narrow chasms [5]. GradienT-RRT can be used in conjunction with GMM to generate more natural motions and can also be used to constrain robot motion using explicit task space constraints. However, this prior work has not considered automatic learning of time-dependent task constraints from demonstrations and using that learned information to guide planning. Learning-based methods can also incorporate consideration of obstacles using potential fields [20], but this approach is sensitive to local minima.

In contrast to prior work, DGMP combines learning from demonstration with a sampling-based motion planner that incrementally refines the solution. Our DGMP framework learns task-space constraints and expresses them in a cost map, considers time-dependent criteria by aligning the robot's trajectory to demonstrations, avoids novel obstacles not included in demonstrations, works for high DOF robots, and generates an optimal solution as computation time is allowed to increase.

3 The Demonstration-Guided Motion Planning Framework

Our Demonstration-Guided Motion Planning (DGMP) framework consists of two major phases: learning and execution. The learning phase only needs to be performed once for a particular task and can then be applied to multiple task executions in different environments. In the DGMP learning phase, human experts perform several demonstrations of the task, which are encoded into a learned cost metric. In the DGMP execution phase, the robot performs the task in a new environment using the derived cost metric for guidance. An overview of the approach is shown in Fig. 2.

The DGMP learning phase requires that human experts control the robot to perform *N* demonstrations of the task. Although the framework allows demonstrations to be provided in any manner, our implementation assumes that we use kinesthetic demonstrations and that the robot includes position-controlled joints. For each

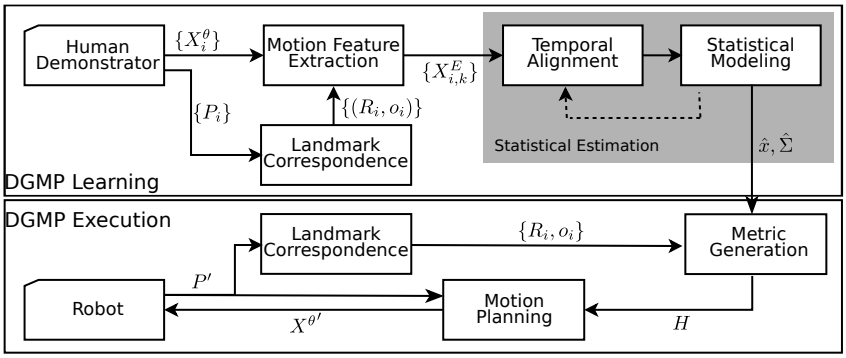

Fig. 2 The DGMP framework consists of a learning phase that is performed once per task and an execution phase that is performed each time the task is executed in a new environment.

demonstration, the robot's joints are placed in a passive mode and a human expert manually moves the robot's limbs to perform the task. We assume the robot has encoders at every joint, allowing the robot to "sense" its own motion and record joint angles as a function of time for each demonstration. The data obtained from demonstration i will be a sequence of joint angles X_i^θ as well as other sensor input P_i of the environment, such as images (from camera sensors) and point cloud data (from laser range finders or stereo image reconstructions).

The DGMP learning phase begins by aligning the demonstrations and transforming the demonstration data into a set of motion features expressed as a function of time. We consider motion features defined by the angles of the robot's joints as well as the position of points on the robot (e.g. the end effector and a grasped object) relative to landmarks in the environment (e.g. the sugar bowl or cup in Fig. 1). For a given time, the mean $\{\hat{x}^{(k)}\}$ and covariance matrix $\{\hat{\Sigma}^{(k)}\}$ of each motion feature k across multiple demonstrations reflect the task constraints intended by the human demonstrator. The lower the variance of a motion feature across demonstrations at a given time, the higher the consistency of the demonstrations with respect to that motion feature. Higher consistency implies the mean value of a motion feature should be followed more closely when performing the task. In contrast, high variance motion features likely do not need to be closely reproduced during execution for the robot to succeed in accomplishing the task.

In the DGMP execution phase, the robot senses its environment to (1) determine the landmarks' correspondences in the current environment, and (2) collect sufficient data to perform collision detection for motion planning. Combining the landmark correspondences with the means $\{\hat{x}^{(k)}\}$ and covariance matrices $\{\hat{\Sigma}^{(k)}\}$ of all the motion features, we define our cost metric H, which estimates the degree to which a a candidate plan matches the intent of the expert demonstrator. We then execute our new motion planning algorithm, MC-RRM, to search for a collision-free robot motion that minimizes the cost metric.

3.1 DGMP Learning Phase

The DGMP learning phase takes as input N demonstrations of the task. During each demonstration i, we record a time sequence of the robot's configuration $X_i^\theta = \{x_{i,t}^\theta\}_{t=1}^{T_i}$, where T_i is the length of the demonstration and $x_{i,t}^\theta$ is the vector joint angles at time t. We also record sensor input $\{P_i, i = 1 \ldots N\}$ of the environment, such as images and point cloud data. From the sensed data, we require that M landmarks be identified, either manually or automatically using computer vision algorithms, that are present across all demonstrations and will likely be present in the execution environment. The landmarks serve as correspondence points in the environment across the demonstrations, and they may or may not be directly related to the task. In our implementation, we also explicitly define the robot's torso to be a landmark. For each demonstration, we denote the poses (orientations and positions) of the landmarks by $\{(R_{ji}, o_{ji}), j = 1 \ldots M, i = 1 \ldots N\}$. For example, in the task shown in Fig. 1 a landmark was sensed on the sugar bowl.

3.1.1 Extracting Motion Features from Demonstrations

We expect that the key task constraints for a problem are satisfied across all the demonstrations. To automatically extract these task constraints, we consider a set of motion features that are designed to help identify aspects of the robot motions that are consistent across demonstrations but that may be hidden in the raw demonstration data. We consider configuration motion features and landmark-based motion features. In a *configuration motion feature*, we consider the robot's joint angles at a particular time. For a personal robot with many redundant degrees of freedom, this data helps in learning "natural" motions that are lost when only considering end effector motions. We denote the configuration motion feature trajectory as $X_i^{(0)} = X_i^{\theta}$. In a *landmark-based motion feature*, we consider the location of one or more points attached to moving parts of the robot (e.g. end effector, manipulator arm, and grasped objects) relative to sensed landmarks in the environment (e.g. a cup or bowl). For the vector of relevant points attached to moving parts of the robot, we define a "local" trajectory of that vector as the coordinates of the relevant points with respect to each landmark. The landmarks essentially serve as local coordinate systems for defining the trajectory of points on the robot. Landmark-based motion features can be used to find important consistencies across demonstrations, such as the position of the end effector relative to a relevant object for the task. We compute the end effector trajectory in the local coordinate system of each landmark, $x_{i,t}^{(j)} = R_{ji}(x_{i,t}^{(1)} - o_{ji}), i = 1 \ldots N, j = 1 \ldots M$, where $x_{i,t}^{(1)}$ is the end effector position relative to the robot's torso (i.e. landmark #1). We represent a local trajectory as $X_i^{(j)} = \{x_{i,t}^{(j)}\}_{t=1}^{T_i}$. Combining the configuration and landmark-based motion features, we have $L = M + 1$ motion feature trajectories represented as $X_i^E = \{X_i^{(j)}\}_{j=0}^{L}$.

3.1.2 Computing the Cost Metric using Statistical Modeling

Our objective is to identify low variance and high variance aspects of the motion feature trajectories across demonstrations in order to create a cost metric that will guide a motion planner to ensure that task constraints are satisfied as best as possible given the locations of obstacles in the execution environments.

The motion feature trajectories are time dependent signals and are obtained from different demonstrations with different time scales (e.g. due to varying demonstration speed). To correctly encode the task constraints across different demonstrations, the trajectories X_i^E must be temporally aligned. That is, for each time step t, the set of motion features for the i'th demonstration $X_{i,t}^E = \{x_{i,t}^{(j)}\}_{j=0}^{M+1}$ is assigned with an aligned time step $g_i(t)$. Therefore, the aligned trajectory of X_i^E is represented as $X_{i,t}^A = \{X_{i,t'}^E | g_i(t') = t, t' = 1 \ldots T_i, i = 1 \ldots M\}$, $t = 1 \ldots T$. We use linear interpolation resampling to ensure one observation per time slot. To compute g_i, we use dynamic time warping (DTW), which has been used in speech recognition [22] and robot learning [9, 12]. DTW uses dynamic programming to compute the optimal time alignment based on a distance function between points. We initially use a Euclidean distance function and then iteratively refine our solution using the output of the statistical modeling as discussed below.

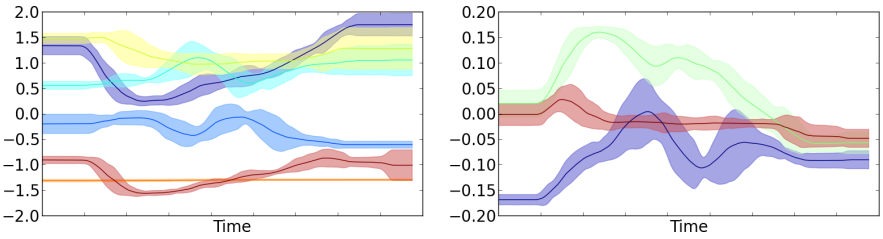

Fig. 3 The mean and covariance of the two motion feature trajectories, the 6D joint angle trajectories (left) and 3D end effector trajectory (right), extracted for the table cleaning task described in Sec. 4. The red line in the right plot, which corresponds to the vertical coordinate of the end effector trajectory, has low variance in the middle section; this indicates the end effector with the paper towel should be kept on the table surface.

Given the time aligned motion feature trajectories, our next objective is statistical modeling: to estimate the expected mean and covariance of the motion features across demonstrations. For a given time t, each motion feature has N observations, one from each demonstration. For notation simplification, let $x_{i,t}^{(k)}$ denote the motion feature k from demonstration i at time t after temporal alignment. We calculate the time dependent mean $\hat{x}$ and covariance matrix $\hat{\Sigma}$ for each time slice:

$$\hat{x}_t^{(k)} = \frac{1}{N}\sum_{i=1}^{N} x_{i,t}^{(k)}, \; \hat{\Sigma}_t^{(k)} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{i,t}^{(k)} - \hat{x}_t^{(k)})(x_{i,t}^{(k)} - \hat{x}_t^{(k)})^T.$$

Fig. 3 shows the statistical modeling result for two motion feature trajectories extracted for the table cleaning task described in Sec. 4.

We use the results of the statistical modeling to iteratively improve the temporal alignment. Rather than treating each DOF of the motion feature trajectories as having equal weight, we instead consider the similarities identified by the covariance computation. We generate weights for each DOF using the inverse of the covariance matrix, which results in similar motion features having a shorter distance in DTW. We re-execute DTW with the revised distance metric, re-evaluate the means and covariances, and loop until the result converges.

In the execution phase described below, we will search for an optimal trajectory in the robot's joint space that follows the modeled constraints, which we represent using a cost metric. The cost metric function depends on the landmark poses in the execution environment, where R_i and o_i define the transformation of the local coordinate system of the i'th landmark. With the input landmark poses provided during task execution, the cost of a given joint configuration θ at time t can be computed by the cost metric H as:

$$H(\theta,t) = (\theta - \hat{\theta}_t)^T W_t^{\theta}(\theta - \hat{\theta}_t) + \sum_{i=1}^{M}[(K(\theta) - \hat{x}_t'^{(i)})^T W_t^{x(i)}(K(\theta) - \hat{x}_t'^{(i)})],$$

where

$$\hat{\theta} = \hat{x}^{(M+1)}, \hat{\Sigma}^{\theta} = \hat{\Sigma}^{(M+1)}, W_t^{\theta} = (\hat{\Sigma}_t^{\theta})^{-1}, \hat{x}_t'^{(i)} = R_i \hat{x}_t^{(i)} + o_i, W_t^{x(i)} = (R\hat{\Sigma}_t^{x(i)} R^T)^{-1}$$

and K is the forward kinematics function mapping the joint configuration to end effector position. The time-dependent cost metric function H is the output of the learning phase.

3.2 DGMP Execution Phase

We use the results of the learning phase to enable the robot to execute the task in new, cluttered environments. The robot must sense its environment to determine the locations of obstacles (e.g. the volume of the workspace that is not free) as well as sense landmarks and establish correspondences to the landmarks used in learning. To compute a motion plan that avoids obstacles while satisfying task constraints, DGMP takes advantage of two useful pieces of information. First, with knowledge of the landmark locations, the robot can compute the learned cost metric defined in the subsection above for any path. Second, it can compute a guiding path defined by the mean of the motion feature trajectories, which is equivalent to following the valley of the cost metric. This guiding path is not guaranteed to be collision free.

To harness these pieces of information and efficiently solve the motion planning problem, we introduce the multi-component rapidly-exploring roadmap (MC-RRM), a sampling based motion planning method that computes a motion plan that minimizes costs over a time-dependent cost map and uses a guiding path to decrease computation time. The guiding path is assumed to follow a valley of the cost map. This method is ideally suited for DGMP by explicitly taking advantage of the information available from the motion feature trajectory means and variances.

MC-RRM combines the benefits of PRMs and RRTs for motion planning problems over time-dependent cost maps in which a guiding path is provided. A traditional PRM with uniform sampling could solve this problem to optimality as the number of samples increases, but this approach is prohibitively slow for high dimensional configuration spaces in which most of the configuration space is not relevant for the task and due to challenges in temporally aligning a PRM plan to the demonstrations for computing the metric. On the other hand, RRT provides a fast incremental sampling-based algorithm for high dimensional spaces but is guaranteed to return a suboptimal solution [15]. MC-RRM builds on ideas from RRG and RRT* [15] and uses the guiding path to incrementally build a roadmap that enables finding an optimal solution as computation time is allowed to increase.

3.2.1 MC-RRM

MC-RRM takes as input a configuration space (C-space), a cost metric over the C-space, a set of obstacles in the C-space, a collision detector, and a guiding path $\tilde{x} = \{\tilde{x}_1 \dots \tilde{x}_T\}$ that traverses a local valley in the cost metric from the initial state to a goal state. The cost metric is defined by $H_e(x_1, x_2) > 0$ for an edge between a pair of arbitrary points x_1 and x_2 in C-space. MC-RRM returns a collision free trajectory such that the sum of the edge costs over the path is minimized.

As shown in Sec. 3.2.2 below, we can compute the guiding path directly based on the cost metric. With the newly observed obstacles in the execution environment, the ideal trajectory defined by this guiding path may not be feasible because parts of the trajectory collide with the obstacles. When the obstacles intercept the trajectory sparsely, the collision-free part of the ideal trajectory is still likely to be the part of the optimal path. Hence, the key challenge is to find optimal alternative pathways for the in-collision sections of the guiding path.

In MC-RRM, the points along the guiding path are regarded as guide points, whether in collision or not. At initialization of the MC-RRM, the collision-free guide points are added as nodes to a global graph (i.e. the roadmap) and adjacent points are connected by an edge if the edge is collision free. Due to obstacles, the global graph may contain multiple disconnected components. Fig. 4 shows an example with two disconnected components. We denoted these components as C_i, which are represented as subgraphs of the global graph. For each C_i, a node set is maintained. The node set is initialized with the guide points from the corresponding disconnected component. In the sampling process, we sample new nodes in a manner similar to RRT for each of the C_i: a random sample is created; nearest neighbors in each C_i are selected; and each component is extended toward the random sample up to a distance $d_{\max}$ where a new node is added. The newly added nodes are connected by edges to the corresponding C_i as well as to the global graph. Every time a new node is added to the global graph, it is connected to all nearby nodes within some distance (as in RRG [15]), which could belong to any C_i. Until the allowed computation is reached, we periodically run Dijkstra's shortest path algorithm to compute the path with lowest cost.

We note that each C_i only maintains a node set and does not need to store edges. Edges are only maintained in the global graph. As the sampling progresses, edges in the global graph may connect nodes in one component with nodes in another disconnected component. We maintain the components separately in order to maintain the expansion bias from each component that was originally separated by an obstacle. This ensures that gaps in the guiding path introduced by obstacles are incrementally explored from multiple directions.

In addition to growing the roadmap from collision-free nodes along the guiding path, we also use a heuristic sampling bias. Rather than sampling uniformly in the robot's configuration space, we consider all the guide points with their covariance matrices as a mixture of Gaussians. We sample configurations from this distribution.

3.2.2 DGMP execution using MC-RRM

MC-RRM requires both a cost metric and a guiding path. For the cost metric, we use H defined in Sec. 3.1.2. For the guiding path, we ignore obstacles and compute the robot configuration (i.e. the joint angles) that minimizes the cost metric at each t, $t = 1 \dots T$. To ensure satisfaction of the robot's kinematic constraints, we minimize H at each t using Lagrange optimization as in Calinon et al. [9]. For the sampling based motion planning algorithm, we compute the edge cost using our cost metric: $H_e(x_1, x_2) = \int_{x_1}^{x_2} H(x, k(x))dx$ where the integral is taken along the line segment from x_1 to x_2 in C-space and k is a time alignment function that estimates the correspond-

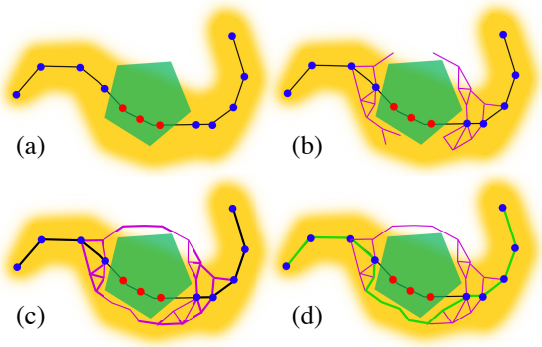

Fig. 4 MC-RRM in a 2D C-space where the guiding path intersects an obstacle (green polygon). (a) The guiding path is added to the graph, resulting in two disconnected components shown in blue. (b,c) Samples are drawn from a GMM (yellow), both components are extended, and nearest nodes are connected (purple), forming a graph. (d) Lowest cost path (green) computed by Dijkstra's algorithm.

ing time index of x in the original demonstration. In our current implementation, k returns the time index of the nearest point on the guiding path. This may introduce problems when the guiding path involves loops, which will be addressed in future work. To efficiently discretize the line integral for $H_e(x_1,x_2)$, we approximate the function as $H_e(x_1,x_2) = \sum_{t=t_1+1}^{t_2} H(\frac{(x_2-x_1)t}{t_2-t_1} + x_1, t)$, where $t_1 = k(x_1), t_2 = k(x_2)$. For the edge from x_1 to x_2, we assume the transition is performed in constant speed and interpolate configurations along the edge.

4 Results

We applied DGMP to the Aldebaran Nao robot [3] to perform two household tasks. The first task was to transfer sugar from a bowl to a cup in the presence of obstacles as shown in Fig. 6 and 7. The second task was to clean the surface of a table in the presence of obstacles as shown in Fig. 9.

The Aldebaran Nao includes 26 total degrees of freedom, including 5 DOF in each arm, 1 DOF for bending at the hip, 1 DOF for opening and closing each 3-finger gripper, and the remaining DOFs for the legs and neck. In our experiments, we used 6 DOF for each task: 5 DOF in the right arm and 1 DOF at the hip. We implemented DGMP using the Python programming language. All computation was performed on a 2.4 GHz Intel Xeon e5620 PC running 64-bit Linux.

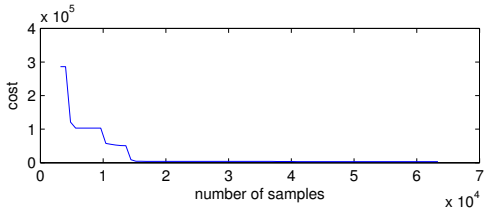

Fig. 5 Minimal path cost vs. number of samples for one of the test cases. With traditional PRM, the minimal cost path is larger than 7×10^6 with 100,000 samples.

In our first experiment, a sugar transfer task, the goal was to evaluate the ability of DGMP to (1) learn task-specific constraints that are critical to the success of the task, and (2) avoid novel obstacles in the environment. The robot was seated at a table with a sugar bowl and a cup, and, in some cases, other items. The task was to scoop some sugar using a spoon and transfer the sugar to the cup without bumping the bowl, the cup, or any other items that may be on the table. Other than the demonstrations, we did not provide the method any explicit task-specific information; e.g. we never explicitly constrained the spoon to be level when moving sugar from the bowl to the cup to prevent spillage. To successfully accomplish this task, the robot needed to (1) automatically learn that the spoon should be kept level as it moves from the bowl to the cup on order to prevent spilling the sugar, and (2) successfully avoid obstacles on the table when performing the task.

We conducted 7 kinesthetic demonstrations. In each demonstration, only the sugar bowl and cup were on the table. We varied the placement of the sugar bowl randomly within a 5 inch radius on the table across the demonstrations. We used 3 environment landmarks which were located on the chair, the cup, and the bowl. Each end-effector trajectory contained the x, y, z coordinates of two points attached to the robot (the end effector and the tip of the spoon), so each end-effector vector is 6 dimensional. Two points on the robot were chosen such that the appropriate tilt and level of the spoon could be learned (via the covariance matrices) for different stages of the task.

To evaluate performance of our method, we created 27 test cases. In each test case, the location of the sugar bowl and of the obstacle were randomly determined. The sugar bowl's location varied within a 4 inch range and the obstacle was placed in a 2.5 inch range. We provided the shape and locations of the bowl and obstacle (as

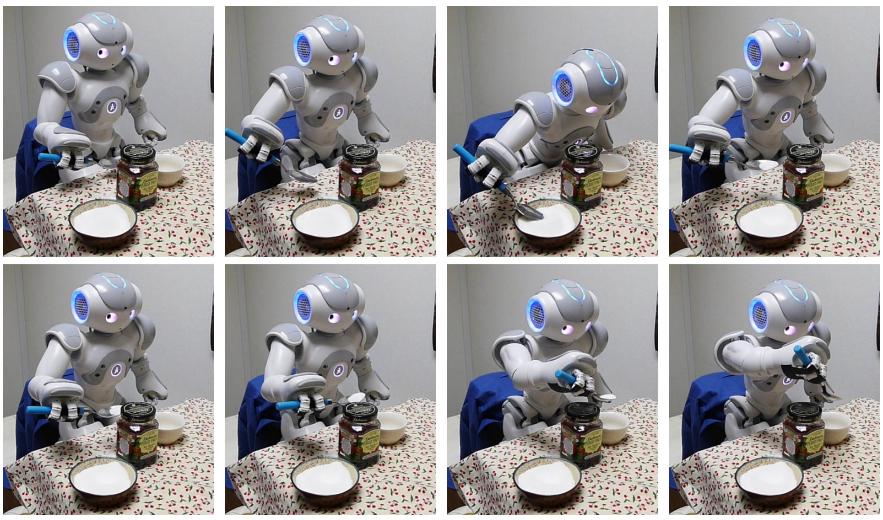

Fig. 6 Execution of DGMP for the sugar transfer task. The robot successfully keeps the spoon level while avoiding the jar, a novel obstacle not included in the demonstrations.

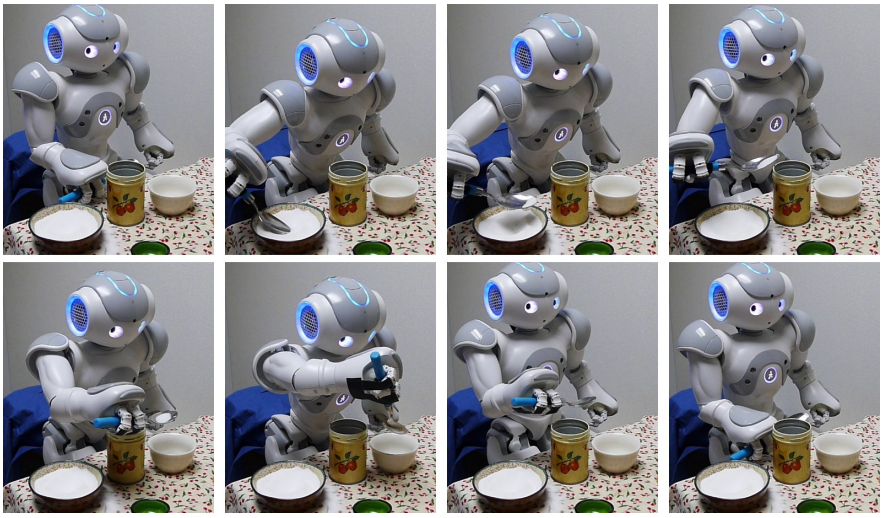

Fig. 7 Execution of DGMP for the sugar transfer task. The robot successfully keeps the spoon level while avoiding the canister, a novel obstacle not included in the demonstrations.

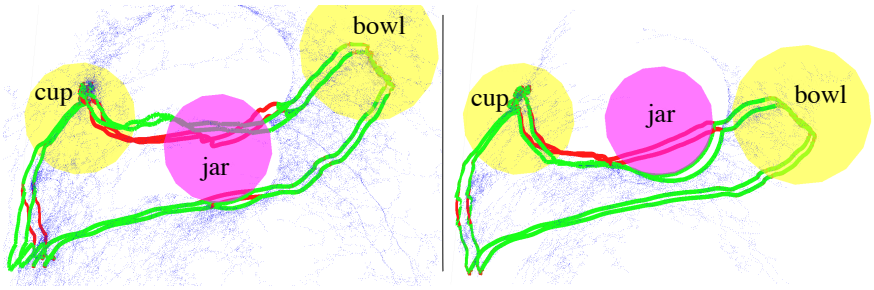

Fig. 8 A top-down view of the workspace for two test cases. The demonstration mean (red) and the MC-RRM plan (green) for the spoon motion, where two points near the spoon tip are tracked. MC-RRM samples projected into the workspace are shown as blue dots. Depending on the placement of the jar, the robot may detour above the jar (left) or to the side of the jar (right) in order to keep the spoon level while still satisfying the kinematic constraints the robot.

would be extracted from a vision system) to the motion planner and then computed a plan for each test case. We then executed each plan on the Nao robot in the experimental setup. After completing the demonstrations, the learning phase required 42 seconds of computation time. In our unoptimized implementation, computing a motion plan then required 1.5 seconds when no obstacle was present and an average of 410 seconds when an obstacle was present. We show the convergence of the method in Fig. 5 and note that the cost metric of the path obtained by MC-RRM was orders of magnitude lower than for standard PRM at equivalent computation times.

A test case was considered *successful* if the robot (1) scoops sugar from the bowl and transfers it to the cup without spilling on the table or obstacle, and (2) does

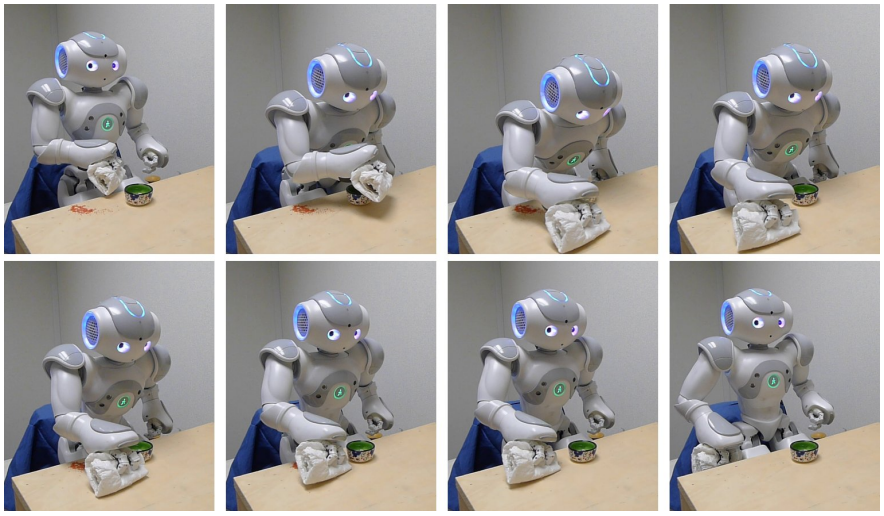

Fig. 9 Execution of DGMP for the table cleaning task. The robot successfully learned to keep the paper towel on the table while avoiding the bowl, which was not included in any of the demonstrations.

not contact the obstacle, bowl, or cup. We considered a test case to be *feasible* if it was possible for the robot to successfully accomplish the task given its kinematic constraints. Of the 27 test cases, 3 were not feasible due to the obstacles being too close to the robot's rest pose.

DGMP succeeded in 22 of the test cases, resulting in a success rate of 92% of the 24 feasible test cases. In the two failure cases, the obstacle was very close to the robot, which results in a narrow passage in the robot's configuration space and the MC-RRM could not find a solution within 100,000 samples. For the same test cases, we also applied a pure learning-based approach in which we executed the mean time-aligned trajectory. Due to obstacle collisions, this approach resulted in only 3 successes, a success rate of 13% of the feasible test cases. RRT resulted in zero successes because it always spills the sugar due to lack of knowledge of task constraints.

We also tested our method on a table cleaning task. As in the sugar transfer task, the Nao robot was seated at a table. The objective was to wipe the table clean using a grasped clump of paper towel. Six demonstrations were recorded of the robot wiping the table in an S-shaped curve without any obstacles. During task execution, a new obstacle was put on the table. To be successful, the robot needed to learn that the paper towel needed to be kept on the surface of the table and avoid obstacles. In all our task executions, DGMP was able to find a detouring path and successfully complete the wiping task, as shown in the example in Fig. 9.

Videos of a Nao robot performing these tasks using DGMP are available at: `http://robotics.cs.unc.edu/DGMP`

5 Conclusion and Future Work

We presented demonstration-guided motion planning (DGMP), a new framework for planning motions for personal robots to perform household tasks. DGMP combines the strengths of sampling-based motion planning and robot learning from demonstrations to generate plans that (1) avoid novel obstacles in cluttered environments, and (2) learn and maintain critical aspects of the motion required to successfully accomplish a task. Sampling-based motion planning methods are highly effective at computing paths from start to goal configurations that avoid obstacles, but task constraints must be explicitly enumerated and programmed. Instead, we used a set of expert demonstrations and automatically extract task constraints by learning low variance aspects of the demonstrations. We then introduced multi-component rapidly-exploring roadmaps (MC-RRM), a sampling-based method that incrementally computes a motion plan that avoids obstacles and optimizes the learned cost metric. We demonstrated the effectiveness of DGMP using the Aldebaran Nao robot performing household tasks in a cluttered environment, including moving a spoon full of sugar from a bowl to a cup and cleaning the surface of a table. By effectively combining motion planning with learning from demonstration, our robot using DGMP accomplished its task successfully in over 90% of its test cases for which a solution was feasible.

In the future work, we plan to extend our method to handle more general tasks with loops and pauses. This could be done by improving the time alignment function in the edge cost metric computation. We also plan to consider real-time sensing (including scene and object recognition) and moving obstacles, which may involve integrating perception-guided motion planning [21] with DGMP. We hope to build on DGMP to enable personal robots to assist humans with a larger class of tasks.

6 Acknowledgement

This research was supported in part by the National Science Foundation (NSF) under awards #IIS-0905344 and #IIS-1117127 and by the National Institutes of Health (NIH) under grant #R21EB011628. The authors thank Jenny Womack from the Dept. of Allied Health Sciences for her input and John Thomas and Herman Towles from the Dept. of Computer Science for their help with the experiment setup.

References

1. P. Abbeel, D. Dolgov, A. Ng, and S. Thrun, “Apprenticeship learning for motion planning with application to parking lot navigation,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sept. 2008, pp. 1083–1090.
2. P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proc. International Conference on Machine Learning (ICML)*, 2004.

3. Aldebaran Robotics, "Aldebran Robotics NAO for education," http://www.aldebaran-robotics.com/en/naoeducation, 2010.
4. B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, pp. 469–483, 2009.
5. 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.
6. A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, ch. 59, pp. 1371–1394.
7. S. Calinon, *Robot Programming by Demonstration*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.
8. 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 *9th IEEE-RAS International Conference on Humanoid Robots*, Dec. 2009, pp. 582–588.
9. S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Systems, Man and Cybernetics–Part B*, vol. 37, no. 2, pp. 286–298, Apr. 2007.
10. H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
11. 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.
12. A. Coates, P. Abbeel, and A. Ng, "Learning for control from multiple demonstrations," in *Proc. 25th International Conference on Machine Learning*, 2008, pp. 144–151.
13. 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.
14. 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.
15. S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robotics: Science and Systems*, 2010.
16. S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.
17. S. J. Lee and Z. Popović, "Learning behavior styles with inverse reinforcement learning," in *ACM Trans. on Graphics (SIGGRAPH 2010)*, vol. 29, no. 4, July 2010, pp. 122:1–122:7.
18. J. Mainprice, E. 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.
19. H. Mayer, I. Nagy, A. Knoll, E. U. Braun, R. Lange, and R. Bauernschmitt, "Adaptive control for human-robot skilltransfer: Trajectory planning based on fluid dynamics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 2007, pp. 1800–1807.
20. P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 763–768.
21. R. B. Rusu, I. A. Sucan, B. P. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 4245–4252.
22. H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.
23. D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *Int. J. Robotics Research*, vol. 29, no. 12, pp. 1565–1592, Oct. 2010.
24. J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2010, pp. 2074–2081.