# Asymptotically Optimal Motion Planning for Learned Tasks Using Time-Dependent Cost Maps

Chris Bowen, *Student Member, IEEE*, Gu Ye, and Ron Alterovitz, *Member, IEEE*

***Abstract*—In unstructured environments in people's homes and workspaces, robots executing a task may need to avoid obstacles while satisfying task motion constraints, e.g., keeping a plate of food level to avoid spills or properly orienting a finger to push a button. We introduce a sampling-based method for computing motion plans that are collision-free and minimize a cost metric that encodes task motion constraints. Our time-dependent cost metric, learned from a set of demonstrations, encodes features of a task's motion that are consistent across the demonstrations and, hence, are likely required to successfully execute the task. Our sampling-based motion planner uses the learned cost metric to compute plans that simultaneously avoid obstacles and satisfy task constraints. The motion planner is asymptotically optimal and minimizes the Mahalanobis distance between the planned trajectory and the distribution of demonstrations in a feature space parameterized by the locations of task-relevant objects. The motion planner also leverages the distribution of the demonstrations to significantly reduce plan computation time. We demonstrate the method's effectiveness and speed using a small humanoid robot performing tasks requiring both obstacle avoidance and satisfaction of learned task constraints.**

***Note to Practitioners*—Motivated by the desire to enable robots to autonomously operate in cluttered home and workplace environments, this paper presents an approach for intuitively training a robot in a manner that enables it to repeat the task in novel scenarios and in the presence of unforeseen obstacles in the environment. Based on user-provided demonstrations of the task, our method learns features of the task that are consistent across the demonstrations and that we expect should be repeated by the robot when performing the task. We next present an efficient algorithm for planning robot motions to perform the task based on the learned features while avoiding obstacles. We demonstrate the effectiveness of our motion planner for scenarios requiring transferring a powder and pushing a button in environments with obstacles, and we plan to extend our results to more complex tasks in the future.**

***Index Terms*—Motion and path planning, assistive robots**

## I. Introduction

ROBOTS have the potential to assist people with a variety of routine tasks in people's homes and workplaces. From assisting a person with a disability with an activity of daily living (such as cooking or cleaning) to assisting a small business owner with a small-scale manufacturing task, assistive robots need to be capable of planning and executing motions in unstructured environments that may contain unforeseen obstacles. Further complicating the planning challenge, many assistive tasks involve significant constraints on motion that humans are aware of from context and intuition. For example, when carrying a plate of food, a person knows that tilting the plate sideways, while feasible, is undesirable because it will spill the food. In order to autonomously and safely accomplish many assistive tasks, a robot must be aware of such task constraints and must plan and execute motions that consider these constraints while avoiding obstacles.

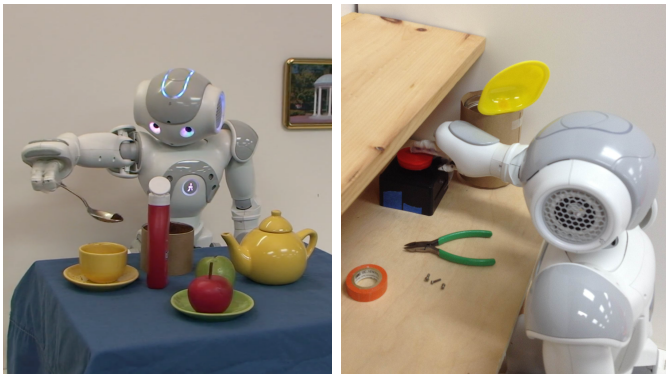


Fig. 1. Tasks in many domains require both avoiding obstacles and also satisfying task constraints. Transferring powder (e.g., instant coffee, sugar, mixes) by spoon requires avoiding obstacles while keeping the spoon level to avoid spills (left). Pushing a (red) button requires that the robot avoid obstacles while ensuring the finger advances at an appropriate orientation when approaching the button (right).

Sampling-based motion planners for robotic manipulators have become highly successful at efficiently computing feasible plans that avoid obstacles [1]. However, these motion planners typically require that the task constraints (such as keeping a plate level) be manually programmed, which can be tedious and requires a programmer with domain knowledge. In contrast, methods based on learning from demonstrations are highly effective at automatically learning task constraints and controllers from demonstrations by people who may not have programming expertise. In this paper, we integrate ideas from demonstration-based learning into a sampling-based motion planner with asymptotic optimality.

We present *demonstration-guided motion planning* (DGMP), a framework for robots to compute motion plans that (1) avoid obstacles in unstructured environments and (2) aim to satisfy learned features of the motion that are required for the task to be successfully accomplished. We focus on tasks in static environments that do not require dynamics and in which task success depends on the relative pose of the robot's end effector to objects in the environment.

At the core of DGMP is an asymptotically optimal sampling-based motion planner that computes motion plans

This research was supported in part by the National Science Foundation (NSF) under awards IIS-0905344, IIS-1117127, and IIS-1149965 and by the National Institutes of Health (NIH) under award R21EB011628. The authors thank Jenny Womack, Pierre Berthet-Rayne, and Wen Sun for their input.

Chris Bowen, Gu Ye, and Ron Alterovitz completed this work while at the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599, USA. Contact e-mail: ron@cs.unc.edu

that are both collision-free and globally minimize a cost metric that encodes learned features of the motion. The motivation for our cost metric is that if the robot is shown multiple demonstrations of a task in various settings, features of the demonstrations that are consistent across all the demonstrations are likely to be critical to task success, while features that vary substantially across the demonstrations are likely unimportant. For example, when transferring instant coffee powder from a container to a cup (see Fig. 1), the feature of the levelness of the spoon will be consistent across the demonstrations (i.e., low variance) while the height of the spoon from the table may vary (i.e., high variance) due to the presence of other objects on the table. Leveraging this insight, our method takes as input a set of kinesthetic demonstrations in which a person holds the robot's limbs and guides the robot to perform the task while we record time-dependent motion features, including the robot's configurations and the pose of the end effector relative to task-relevant objects in the environment. The placement of these objects, such as the coffee container and cup in the example above, are randomized for each demonstration. We then apply statistical approaches to compute a learned task model that encodes the covariances of the motion features as a function of time.

Once the task model is learned, DGMP can be used to autonomously execute the learned task in a static environment in which task-relevant objects may be in different locations and new obstacles may be present. Using the learned task model, we define a time-dependent cost map specific to the current environment. The cost map, defined over the robot's configuration space, considers the covariances of the motion features across demonstrations using a Mahalanobis distance metric and is parameterized by the locations of the task-relevant objects. The cost map is defined such that a trajectory that minimizes cost has the highest probability of coming from the distribution of the demonstrations in motion feature space. We then introduce DGPRM (demonstration-guided probabilistic roadmap), an asymptotically optimal sampling-based motion planner that minimizes the integral of the learned cost map along the planned trajectory.

The novelty of DGPRM comes from the fact that it combines into a single motion planner multiple useful properties when utilizing demonstration-based information. First, DGPRM considers time-dependent cost maps, which is important since task constraints vary over the duration of a motion and tasks may require being at the same configuration at different time points of the task. Second, DGPRM provides asymptotic optimality, meaning it finds a trajectory that avoids obstacles in a manner that is globally optimal with respect to the learned cost metric. Third, DGPRM introduces speedups that use the distribution of demonstrations inherent to the learned task model to significantly accelerate motion planning.

In this work, we provide a refined version of the results presented at a conference [2] and incorporate several important extensions. We also show the effectiveness of DGMP using the Aldebaran NAO small humanoid robot [3] performing assistive tasks in environments with never-before-seen obstacles.

## 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 [1], [4]. While most sampling-based motion planners aim to minimize metrics such as Euclidean distance in the workspace or configuration space, some methods have investigated incorporating task constraints. Several approaches are based on rapidly exploring random trees (RRTs) [1], a highly successful method for computing feasible, obstacle-avoiding trajectories but which does not guarantee plan optimality [4]. 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. Algorithms include sampling only inside a user-specified number of standard deviations of a mean demonstrated trajectory [9], finding low-cost paths over cost maps using local optimization [10], locally optimizing a specified objective function using gradient descent [11], and enforcing constraints using sampling strategies [12]. Prior sampling-based motion planning approaches, unlike our proposed method, do not simultaneously guarantee asymptotic optimality and allow for time-dependent task constraints.

At the heart of our method is 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 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 [1] or sampling-based planners not designed for asymptotic optimality like RRT [4]. Related work has investigated asymptotically optimal planners that balance exploration and refinement [13], asymptotic near-optimal planners using smaller roadmaps [14], and anytime solution optimization [15]. Our method integrates a learned cost metric with RRG or a PRM variant [4] to guarantee asymptotic optimality for our learned cost metric.

Our method combines a new sampling-based motion planner with ideas from demonstration-based learning, which has been highly successful in enabling robots to learn task constraints and imitate task motions [16], [17]. Our focus is not on learning control policies for dynamic systems (e.g., [18], [19], [20], [21]) but rather on computing robot trajectories that avoid obstacles while satisfying learned constraints. Our aim is globally optimal obstacle avoidance in which the robot considers plans in all homotopic classes and selects the best one. Prior work has investigated using search methods such as A* or D* where cost maps or movement costs are learned from demonstrations (e.g., [22], [23], [24], [25]), which are highly effective for 2D, discrete state spaces but do not scale

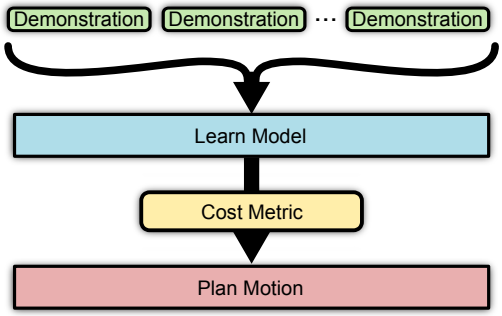



Fig. 2. The DGMP framework consists of a learning phase, which is performed once per task, and an execution phase, which involves planning the motion and is performed each time the task is executed in a new environment.

well to higher degree of freedom systems like robotic arms.

An alternative approach is to locally avoid obstacles, which works well for some applications but does not guarantee global optimality. Potential field approaches have been applied to dynamic movement primitives [26] and a Gaussian mixture model (GMM) [27] to locally avoid obstacles, but potential fields require setting parameters for obstacle repulsion and can result in a robot being trapped in local minima, especially in obstacle concavities or narrow passages [1].

Another approach is to include the obstacles in the demonstrations. Calinon et al. introduced a GMM and Gaussian mixture regression (GMR) approach to learn motions relative to task-relevant objects and obstacles that are present in both the demonstration and execution environments [28], [29]. This approach represents a task using a hidden Markov model (HMM); HMM's have been used for motion recognition (e.g., [30], [31], [32]) and generation (e.g., [28], [32]).

We also use an HMM; however, we use a restricted form which permits us to build on prior work on dynamic time-warping [18], [19] to create high quality alignments using an expectation-maximization approach and then directly compute means and covariances in the space of motion features. We use this model to construct a time-dependent cost map which we can integrate into an asymptotically optimal sampling-based motion planner for obstacle avoidance.

## III. Method Overview

Let $\mathcal{Q} = \mathbb{R}^d$ be the $d$-dimensional configuration space of the robot and $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ denote the set of configurations in which the robot is not in collision with an obstacle. We assume the robot has also sensed the poses of $L$ task-relevant objects (such as the cup and instant coffee container in Fig. 1(left)), which are stored in a vector $\mathbf{a} \in \text{SE}(3)^L$, and that these objects remain stationary as the task is performed. We also assume the robot is holonomic with position-controlled joints, and we do not consider dynamics. Our objective is to compute a trajectory $\Phi \in [0,1] \rightarrow \mathcal{Q}_{\text{free}}$ from the robot's initial 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 robot successfully accomplishes the task.

To address this challenge, we develop an approach for demonstration-guided motion planning (DGMP) that consists of two major phases: learning and execution. Fig. 2 illustrates an overview of the approach. The approach requires as input a set of user-provided demonstrations of the task. During the cost metric learning phase, the robot learns from the demonstrations a time-dependent cost metric for the task that considers the robot's configuration and its motion relative to task-relevant objects. The learning phase need only be performed once per task. When the robot is in a new environment, the robot enters the motion planning phase in which it computes a path that minimizes the learned cost metric, which captures aspects of the demonstrated motions that are required to perform the task.

During the DGMP learning phase, presented in Sec. IV, we first extract from each demonstration a set of *motion features* that quantify properties of the motion as a function of time, such as joint angles or the location of the end effector with respect to a task-relevant object. After time-aligning the demonstrations, we compute statistics on the motion features, including their means and variances, over time across the demonstrations. 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 feature, which 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.

To compute a cost metric, we will leverage the intuition above regarding the increased importance of motion features with low variances. Formally, we consider the demonstrations (as encoded in the space of motion features) to be samples from the distribution of trajectories that will succeed at the task. We then model the quality of a candidate motion plan in the execution environment as the likelihood that it, too, is a sample from this distribution of successful trajectories and define a cost metric such that better plans have lower costs.

In the DGMP execution phase, presented in Sec. V, the robot first senses its environment to collect sufficient information to evaluate the learned cost metric and to perform collision detection. We then execute our new asymptotically optimal motion planning algorithm, DGPRM, to search for a feasible, collision-free motion plan that minimizes the learned cost metric, and hence reproduces the demonstrator's intent as closely as possible in the new environment.

## IV. Learning the Time-Dependent Cost Metric

In the first phase of DGMP, we learn a time-dependent cost metric for a task based on the robot's configurations and motions relative to task-relevant objects in a set of demonstrations. The cost metric encodes the spatial variations and temporal ordering of the task. The robot will later use this cost metric when planning its motions to complete the task in new environments where task-relevant objects may have moved and new obstacles may be present.

Rather than directly learning constraints, we learn a cost metric, an approach that offers two advantages. First, a cost metric better models how we observe humans perform a task. A human holding a spoon to transfer powder typically holds the spoon roughly level with no explicit, hard bounds on deviations from level. Second, the use of a cost function

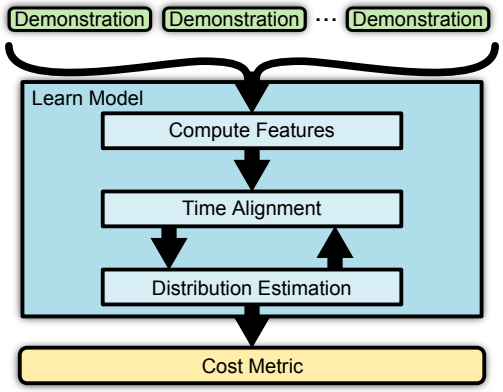



Fig. 3. The DGMP learning phase lifts demonstrations into a feature space, iteratively time aligns the demonstrations, and learns a cost metric in the feature space.

allows us to learn relatively complex tasks from a small number of demonstrations. While it is hard to differentiate the relevant similarities across demonstrations from the infinitely many irrelevant coincidences without the extensive semantic knowledge that a human would have, it is relatively easy to measure similarity between a candidate trajectory and the demonstrations (as is required for cost metric learning). We show an overview of the learning phase in Fig. 3.

### A. Inputs and Outputs of Cost Metric Learning

As input to the DGMP learning phase, a human controls the robot to perform $M$ demonstrations of the task. For each demonstration, the robot's joints are placed in a passive mode while a human manually moves the robot's limbs to perform the task and indicates where the task begins and ends. We assume the robot has encoders at every joint, allowing the robot to sense its own motion and record its configuration (e.g., a vector of its joint angles) as a function of time. During each demonstration $m \in \{1, \ldots, M\}$, we record a time sequence of the robot's configuration $\{\mathbf{q}_{m,s}\}_{s=1}^{S_m}$, where $S_m$ is the length of the demonstration and $\mathbf{q}_{m,s}$ is the configuration at time $s$. For each demonstration, we also require an annotation $\mathbf{a}_m \in \mathrm{SE}(3)^L$ identifying the poses of $L$ task-relevant objects in the environment (e.g., the coffee container and cup in the task shown in Fig. 1), which could be identified either manually by the human or automatically using computer vision algorithms. We denote the poses in demonstration $m$ of the task-relevant objects $\{(\mathbf{R}_{m,l}, \mathbf{o}_{m,l}) \mid m = 1, \ldots, M; l = 1, \ldots, L\}$, where $\mathbf{R}_{m,l}$ is the rotation matrix and $\mathbf{o}_{m,l}$ is the translation vector of landmark $l$ with respect to a global frame. Because the objects are task-relevant, they should be present in all demonstrations and necessarily must be present in the execution environment.

The output of the DGMP learning phase is a time dependent cost metric $c(\mathbf{q}, t, \mathbf{a})$ for robot configuration $\mathbf{q}$ at time $t$ for an execution environment with annotations $\mathbf{a}$, which may be different from any of the values of $\mathbf{a}$ in the demonstrations.

### B. Extracting Motion Features from Demonstrations

Since each demonstration is a successful task execution, we expect that the task constraints for a problem are satisfied in each demonstration. To enable learning of 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. These motion features may depend on both the configuration and annotation. We denote motion feature $j$ for time step $s$ of demonstration $m$ as $\mathbf{y}_{m,s}^{(j)}$. Inspired by results from Calinon et al. [28], [29], for our experiments we consider two classes of motion features:

- A *configuration motion feature* is the robot's configuration at a particular time. When there are redundant degrees of freedom, this data enables learning natural motions that are lost when only considering end-effector motions. We define this motion feature as
$$\mathbf{y}_{m,s}^{(0)} = \mathbf{q}_{m,s}.$$
- A *landmark-based motion feature* is a vector of the coordinate of a point on the robot $\mathbf{x}'$ (e.g., end-effector, grasped object) relative to a landmark on a task-relevant object in the environment (e.g., cup, button). This motion feature facilitates task execution for cases in which task-relevant objects may be located in different places across demonstrations and during execution. We define the motion feature relative to landmark $l$ as
$$\mathbf{y}_{m,s}^{(l)} = \mathbf{R}_{m,l}^{-1}(\mathbf{x}'_{m,s} - \mathbf{o}_{m,l}).$$

Because we compute these motion features from the same information, namely the configuration and annotation, it is convenient to consider both types of motion features in the context of a unifying joint motion feature space $\mathcal{Y}$ along with some general function $f \in (\mathcal{Q}, \mathrm{SE}(3)^L) \to \mathcal{Y}$ which lifts a configuration $\mathbf{q}$ into the motion feature space given some annotation $\mathbf{a}$. Such a function can represent multiple motion features simply by computing each motion feature $\mathbf{y}^{(j)}$ individually and concatenating them into a single higher-dimensional motion feature vector $\mathbf{y} = f(\mathbf{q}, \mathbf{a})$.

Similarly, we may consider a trajectory of motion features for each demonstration which we denote by $Y_m = \{\mathbf{y}_{m,1}, \mathbf{y}_{m,2}, \ldots, \mathbf{y}_{m,S_m}\}$ where $\mathbf{y}_{m,s} = f(\mathbf{q}_{m,s}, \mathbf{a}_m)$. Constructing these motion features is the sole purpose of the demonstrations. The remainder of the cost metric learning method operates exclusively in motion feature space.

### C. Statistical Modeling of the Motion Features

Our objective is to identify consistent aspects of the motion feature trajectories across demonstrations in order to create a cost metric, parameterized by the locations of the task-relevant objects, which will guide the motion planner.

To achieve this, we learn a statistical model consisting of $T$ multivariate Gaussian distributions $\mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$ in motion feature space, each of which models the distribution of motion feature vectors of the demonstrations at some time step $t \in \{1, \ldots, T\}$ in the task. Hence, our model of the task is parameterized by the mean $\mu_t$ and the covariance matrix $\mathbf{\Sigma}_t$ of motion feature vectors for each time step $t$. These may be considered as output distributions in a simple HMM with $T$ sequential states, wherein each state $t$ has nonzero and equal transition probabilities only to itself and the next

state $t+1$. This induces a linear order structure to the HMM corresponding to time. The problem then is to find the output distributions most likely to have generated the demonstrations.

To learn these distributions, we must find the correct monotonic mapping, or *alignment*, between each of the observed configurations in the demonstrations and the $T$ time steps. This corresponds to determining the walk in the HMM which generated the demonstration. This is necessary because the demonstrations are of different lengths and may perform parts of the task more or less quickly. For instance, the demonstrations of the powder transfer task mentioned previously varied from 21 to 32 seconds in length. Estimating to which state of the task a given observed configuration in a demonstration corresponds requires constructing a model of the task, which is exactly why we needed such an alignment in the first place. We resolve this cyclic dependence by applying an expectation-maximization (EM) algorithm, a common approach to learning models of processes with latent variables.

First we choose a random initial alignment for each demonstration to the time steps in the task. Next we estimate each distribution $\mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$ using the sample mean and covariance of the motion feature vectors which are aligned to time step $t$ (weighted inversely proportional to the number of observed configurations aligned to time step $t$ from the same demonstration). This is the E step of the EM algorithm. Next we find, for each demonstration, the walk in the estimated HMM most likely to have generated the demonstration using dynamic time warping (DTW) [33]. This is the M step of the EM algorithm. Finally, if the algorithm has not yet converged, we go back to the E step. Because EM algorithms can become caught in local minima, we repeat the entire method a number of times with different randomized initial alignments and use the resulting alignment with maximum likelihood.

More formally, let $Y_{m,t}$ denote the set of motion feature vectors in demonstration $m$ which are aligned to time step $t$. The E step computes the weighted sample mean and covariance at each time step from $Y_{m,t}$ as follows:

$$w_m^{(t)} \leftarrow \frac{1}{|Y_{m,t}|}, \quad \mu_t \leftarrow \frac{1}{M}\sum_{m=1}^{M}\left(w_m^{(t)}\sum_{\mathbf{y}\in Y_{m,t}}\mathbf{y}\right),$$

$$\mathbf{\Sigma}_t \leftarrow \frac{M}{M^2-\sum w_m^{(t)}}\sum_{m=1}^{M}\left(w_m^{(t)}\sum_{\mathbf{y}\in Y_{m,t}}(\mathbf{y}-\mu_t)(\mathbf{y}-\mu_t)^T\right).$$

The M step uses DTW to align each of the demonstrations to the distributions learned in the E step, using a cost function which maximizes the likelihood that the motion feature vectors aligned to time step $t$ in the demonstration came from the distribution $\mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$. The formulation of this cost is discussed in more detail in Sec. IV-D. Our results show that using an EM approach, rather than only DTW as is commonly done in prior work, is crucial for effectively learning the task.

Accurately estimating the covariance matrices for the Gaussian distribution at each time step requires that we have a sufficient number of demonstrations. The number of demonstrations should exceed the dimension of the motion feature space. Intuitively, if the number of the demonstrations is smaller, then one or more time steps could have too few motion feature vectors aligned to it, resulting in a singular matrix. In this case, we are only learning in a subspace of the motion feature space. We note that this lower bound is empirically tight for some problems, as shown in the results section.

It is possible to reduce the number of required demonstrations by approximating the motion features as independent. This corresponds to computing the covariance matrix of each configuration or landmark-based motion feature independently, resulting in $\mathbf{\Sigma}_t$ being a block diagonal matrix. This reduces the number of required demonstrations to be one plus the dimension of the largest motion feature vector. We used this approach in our experiments with the NAO robot in Sec.VI and were able to effectively capture relevant task constraints.

### D. Cost Metric

To formally define the cost metric, we consider the demonstrations to be samples from the distribution of trajectories that will succeed at the task. Given an annotation $\hat{\mathbf{a}}$ for the environment, we define the cost of a candidate trajectory in the environment based on how likely it is a sample from the distribution of successful trajectories.

With this approach, we wish the probability of the trajectory being generated by the task model to be maximized when the cost metric is minimized. At a given time step $t$, with a configuration $\mathbf{q}$ this probability is given by

$$\mathrm{P}(\mathbf{q}, t \mid \mu_t, \mathbf{\Sigma}_t, \hat{\mathbf{a}}) = N_{\mathbf{\Sigma}_t} e^{-\frac{1}{2}(f(\mathbf{q},\hat{\mathbf{a}})-\mu_t)^T\mathbf{\Sigma}_t^{-1}(f(\mathbf{q},\hat{\mathbf{a}})-\mu_t)}$$

where $N_{\mathbf{\Sigma}_t}$ is a normalization factor. However, operating in the space of probabilities is numerically inconvenient, so we instead consider the log probability as is common practice in the machine learning literature, yielding

$$\log \mathrm{P}(\mathbf{q}, t \mid \mu_t, \mathbf{\Sigma}_t, \hat{\mathbf{a}}) = \log(N_{\mathbf{\Sigma}_t}) - \frac{1}{2}(f(\mathbf{q},\hat{\mathbf{a}})-\mu_t)^T\mathbf{\Sigma}_t^{-1}(f(\mathbf{q},\hat{\mathbf{a}})-\mu_t).$$

When performing time alignment, it is this value which we maximize in the M step, but in the case of the cost metric, it can be simplified further. First by observing that the log normalization term is constant for a given time step and thus has constant contribution to the total cost of a trajectory, we can safely ignore it. Finally, we drop the $-\frac{1}{2}$ constant factor. This changes the sign, which is desirable because we wish to formulate the problem as a minimization rather than a maximization. The final cost map is thus

$$c(\mathbf{q}, t, \hat{\mathbf{a}}) = (f(\mathbf{q},\hat{\mathbf{a}})-\mu_t)^T\mathbf{\Sigma}_t^{-1}(f(\mathbf{q},\hat{\mathbf{a}})-\mu_t). \quad (1)$$

We note that this is simply the squared Mahalanobis distance [34] in feature space from $\mathbf{q}$ to the configurations observed in the demonstrations at time $t$. The cost metric we will minimize is the integral over this cost map.

In the following discussion we will drop the dependence on $\mu$, $\mathbf{\Sigma}$, and $\hat{\mathbf{a}}$ from the notation for the sake of brevity.

This log probability formulation has desirable properties, the most notable of which is composition. The sum of the log probabilities is the log of the product of the probabilities, $\log \mathrm{P}(\mathbf{q}, t) + \log \mathrm{P}(\mathbf{q}', t') = \log(\mathrm{P}(\mathbf{q}, t)\mathrm{P}(\mathbf{q}', t'))$. We first assume that $\mathbf{q}$ and $\mathbf{q}'$ are independent as will be the case when

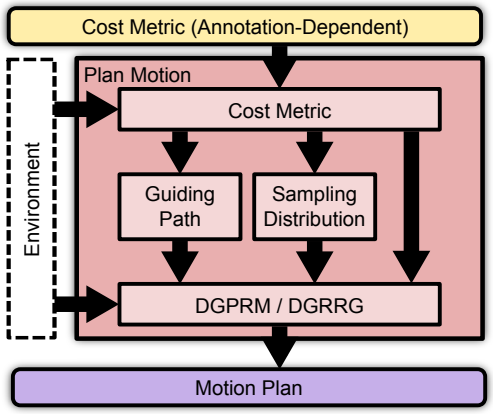



Fig. 4. The DGMP execution phase uses output from the learning phase to construct a cost metric for the current execution evironment and then computes a motion plan that minimizes the cost metric.

they are drawn independently from a sampling distribution by a sampling-based motion planner. We then assume independence between time steps, which while not generally true, is a convenient simplifying assumption. Under these independence assumptions, $\mathrm{P}(\mathbf{q}, t)\mathrm{P}(\mathbf{q}', t')$ is the joint probability given the task model, so $\log \mathrm{P}(\mathbf{q}, t) + \log \mathrm{P}(\mathbf{q}', t') = \log \mathrm{P}(\mathbf{q}, t, \mathbf{q}', t')$. This is important because our motion planner will find a trajectory which minimizes the integral of this cost map, which under these assumptions is equivalent to maximizing the probability of the entire trajectory in our learned model.

We note that computing the likelihood given the model requires that we compute $\mathbf{\Sigma}_t^{-1}$, which exists only if $\mathbf{\Sigma}_t$ is non-singular. Cases where $\mathbf{\Sigma}_t$ is singular will arise, e.g., when multiple landmarks are fixed relative to each other. To overcome this in our implementation we employ a pseudoinverse, specifically the Moore-Penrose pseudoinverse. This approach has the desirable property of effectively collapsing multiple landmarks fixed relative to each other into a single landmark.

The learned cost map depends on knowing $\hat{\mathbf{a}}$, which contains the locations of the task-relevant objects. Hence, the cost metric is used after the robot senses the locations of the task-relevant objects in the execution environment. We describe in Sec. V how the learned cost metric is used in motion planning.

## V. Motion Planning Using the Learned Cost Metric

In the DGMP execution phase, the robot computes a feasible, collision-free motion plan in configuration space that minimizes the learned cost metric. We show an overview of the execution phase in Fig. 4.

### A. Inputs and Outputs of Motion Planning

The DGMP execution phase requires as input an annotation $\hat{\mathbf{a}}$ describing the new environment and a model of the obstacles that must be avoided. In our experiments, we used color and depth data from a Microsoft Kinect to automatically create models of obstacles as described in Sec. VI. The method also requires as input the robot's start and goal configurations $\mathbf{q}_{\mathrm{start}}, \mathbf{q}_{\mathrm{goal}} \in \mathcal{Q}_{\mathrm{free}}$ and the time-dependent cost map $c \in (\mathcal{Q}, [0,1]) \rightarrow \mathbb{R}^+$ as given by Eq. 1. For notational convenience, we scale time to be between 0 and 1 and drop the annotation parameter from $c$ since for motion planning this is always the observed execution environment $\hat{\mathbf{a}}$.

We say $\Phi \in [0,1] \rightarrow \mathcal{Q}$ is a *trajectory* if and only if it is Lipschitz continuous. That is $\exists K_\Phi \in \mathbb{R}^+, \quad \forall t_1, t_2 \in [0,1], \quad |\Phi(t_2) - \Phi(t_1)| < K_\Phi(|t_2 - t_1|)$. We say a trajectory $\Phi$ is *feasible* if and only if $\forall t \in [0,1], \quad \Phi(t) \in \mathcal{Q}_{\mathrm{free}}$, $\Phi(0) = \mathbf{q}_{\mathrm{start}}$, and $\Phi(1) = \mathbf{q}_{\mathrm{goal}}$. Let $C(\Phi) = \int_0^1 c(\Phi(t), t)\,\mathrm{d}t$ denote the cost of a trajectory $\Phi$. As discussed previously, our choice of cost metric has the advantageous property that the sum of the costs is the log likelihood in the joint distribution under the assumption of independent time steps. This is the discrete analogue of the integral formulation of $C$ used by the motion planner.

Our objective is to compute a feasible trajectory $\Phi^*$ that minimizes cost $C(\Phi^*)$.

### B. Sampling-Based Planning for the Learned Cost Metric

We introduce DGPRM, a new sampling-based motion planner for computing plans that minimize the DGMP time-dependent cost metric. We employ a variation of a probabilistic roadmap (PRM) [35], because of its asymptotic optimality (using the sPRM variant) [4] and ease of parallelization, which we leverage. We also integrate DGMP with an RRG-based roadmap [4], which performs roughly equivalently when used with our DGMP-based extensions as discussed in the results. Our sampling-based motion planner guarantees that, as computation time is allowed to increase, all homotopic classes of plans will be considered and an optimal plan approached.

PRM methods construct a graph (called a roadmap) where each vertex (called a waypoint) corresponds to a configuration of the robot and each edge corresponds to a local plan for navigating from the configuration of one waypoint to another. This graph is constructed by repeatedly sampling a configuration from $\mathcal{Q}$ and adding a new waypoint to the roadmap corresponding to this configuration if it is collision-free. When a new waypoint is added to the roadmap, edges are constructed between it and other waypoints which are nearby in configuration space and connectable by collision-free paths. As the number of waypoints currently in the roadmap increases, the roadmap becomes a denser approximation of the collision-free configuration space.

To accommodate the time-dependency in the cost metric, we associate a time value with each waypoint and use a directed graph for the roadmap to forbid traversing edges backwards in time. We choose the time value associated with a given waypoint by maintaining a partitioning of the time span $[0,1]$ which is initially just a single partition consisting of the entire time span $T = \{[0,1]\}$. These partitions can be thought of as *layers* within the roadmap. We also choose some initial value $\Delta\mathbf{q}_{\max}$. We then alternate between two phases, *expansion* and *splitting*. Intuitively, these increase sampling density in configuration space and in time respectively. Throughout the process, we track the size of the largest partition, which we denote $\Delta t_{\max}$.

This iterative refinement of time partitions provides multiple benefits. The first and most notable of which is the

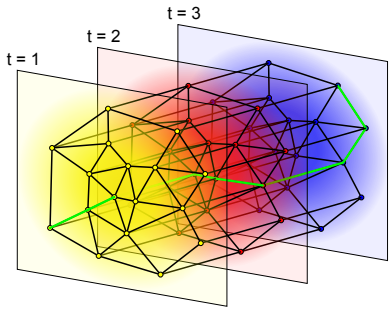



Fig. 5. An example roadmap for a 2D configuration space with 3 time partitions. A path is shown in green.

capability to handle time-dependent cost metrics as present in DGMP. The second benefit of this approach is that we do not require numeric integration of cost along roadmap edges, which typically requires an integration step size parameter. This approach can result in "missing" a small, high-cost region if this parameter is not properly tuned. In contrast, as DGPRM progresses, the maximum step size in both configuration space and time automatically decrease in such a way that the error in computing the cost metric for a trajectory approaches 0 under certain reasonable assumptions (see Sec. V-D).

DGPRM begins by adding the start and goal configurations to the roadmap. Due to the layers, multiple waypoints may correspond to the same configuration. We say that the start waypoint is the waypoint corresponding to the start configuration in the first time partition, and the goal waypoint is the waypoint corresponding to the goal configuration in the last time partition. We let $N$ denote the number of configurations currently in the roadmap, so initially $N = 2$.

At any time, we may search the roadmap for the shortest path from the start waypoint to the goal waypoint. Interpolating linearly along this path yields a feasible trajectory, which is taken as an approximation of $\Phi^*$. An illustrative roadmap is shown in Fig. 5.

*1) Expansion:* In the expansion phase, $\Delta N$ additional configurations are sampled from $\mathcal{Q}$. For each configuration, if the configuration is collision free we add a waypoint to the roadmap in each time partition for the configuration. As in other PRM methods, edges lying entirely in $\mathcal{Q}_{\text{free}}$ are then added between nearby waypoints. In this case, nearby means nearby in both space and time. Specifically, edges are only added if the waypoint configurations are within $\Delta\mathbf{q}_{\max}$ and the time values are within adjacent partitions. The edges are also directional, and oriented forward in time.

*2) Splitting:* In the splitting phase, we first choose a time value $t$ at which to split. The only restriction on this choice is that to approach optimality in the limit, the size of every partition must approach 0. That is, $\Delta t_{\max} \to 0$. To perform the split, all the vertices in the time partition containing $t$ are duplicated, including their incoming and outgoing edges. All of these vertices are then assigned new time values in one of the two new partitions. Next, $\Delta\mathbf{q}_{\max}$ is updated based on the new value of $\Delta t_{\max}$, and edges longer than $\Delta\mathbf{q}_{\max}$ or that span multiple time partitions are pruned from the roadmap. Edges which violate the temporal ordering property are reoriented. Finally, all affected edge costs are recomputed.

This method frequently requires that we measure distance in configuration space, both for connecting nearby waypoints and for pruning long edges. While any distance metric could be used here, a metric which better approximates the actual cost of traversing an edge improves performance. In DGMP we use the Mahalanobis distance in motion feature space with the covariance matrix of all the motion feature vectors from all the demonstrations.

### C. Demonstration-Guided Speedups for Motion Planning

While not strictly necessary for motion planning, we compute a *guiding path*, the trajectory which minimizes $C$ in the absence of obstacles. The configuration at time $t$ along the guiding path is given by

$$\hat{\mathbf{q}}_t = \underset{\mathbf{q}\in\mathcal{Q}}{\operatorname{argmin}} \, (f(\mathbf{q}, \hat{\mathbf{a}}) - \mu_t)^T \boldsymbol{\Sigma}_t^{-1} (f(\mathbf{q}, \hat{\mathbf{a}}) - \mu_t). \quad (2)$$

Efficient methods exist for locally approximating this computation [36]. Computing the guiding path is fast and can be used to speed up computation time in several ways.

First, we implemented *seeding* which adds configurations along the guiding path to the initial roadmap. These waypoints serve as local minima in regions of configuration space that are collision-free.

Second, we use the guiding path to *bias* configuration space sampling to reduce computation time in environments with obstacles. In the expansion step of DGPRM, rather than using uniform sampling from $\mathcal{Q}$, we sample configurations based on the learned model, sampling more densely in regions of the configuration space that are more likely to result in high quality plans. Specifically, we sample $\mathbf{q}$ from a Gaussian in configuration space with mean at the guiding path configuration at that time step $\hat{\mathbf{q}}_t$. The covariance matrix is chosen to be the sample covariance of all the configurations across all the demonstrations.

In addition to the speedups based on the demonstrations, we note that our use of layers provides a speedup for problems in static environments with time dependence. When a configuration is added to the roadmap, we add corresponding waypoints and edges to all layers but only need to check for collisions once, which reduces computation time compared to prior approaches that sample in the product of configuration space and time and require collision checking for each sample.

### D. Analysis

In this section, we will provide an outline of a proof showing that, under certain assumptions and for suitable choices of parameters, the method is guaranteed to converge to an optimal trajectory with probability 1 as computation time increases. For the full proofs of the lemmas and theorem below, see the included supplementary material (also available at http://robotics.cs.unc.edu/DGMP2).

To simplify the analysis, we consider a modified version of the method in which a new roadmap is constructed at each iteration and time partitions are evenly distributed. In this section we assume that a minimal feasible trajectory $\Phi^*$ exists.

**Assumption 1.** *The cost map $c$ is Lipschitz continuous.*
$\exists K_c \in \mathbb{R}^+, \quad \forall \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{Q}, \quad \forall t_1, t_2 \in [0,1],$
$|c(\mathbf{q}_2, t_2) - c(\mathbf{q}_1, t_1)| < K_c \max(|\mathbf{q}_1 - \mathbf{q}_2|, |t_2 - t_1|).$

We note that the DGMP cost map satisfies this assumption if the function $f$ which lifts from configuration space to feature space is Lipschitz continuous and none of the learned covariance matrices are singular.

Where $\pi = (\mathbf{q}_1, t_1, \ldots, \mathbf{q}_k, t_k)$ is a path in the roadmap defined by the sequence of configurations $(\mathbf{q}_i)$ at times $(t_i)$, let $\Phi_\pi$ be the trajectory constructed by linearly interpolating between configurations in $\pi$. Specifically, let $\Phi_\pi(t) = \mathbf{q}_i + \frac{t - t_i}{t_{i+1} - t_i}(\mathbf{q}_{i+1} - \mathbf{q}_i)$ where $t_i \leq t \leq t_{i+1}$. By construction of the roadmap, all $t_i$ are distinct and $\forall i < k, \quad |\mathbf{q}_{i+1} - \mathbf{q}_i| < \Delta\mathbf{q}_{\max}$ so $\Phi_\pi$ is Lipschitz continuous with $K_\Phi = \frac{\Delta\mathbf{q}_{\max}}{\min\limits_{1 \leq i < k} t_{i+1} - t_i}$. Furthermore, edges are only added to the roadmap if the line segment between the waypoints is contained in $\mathcal{Q}_{\text{free}}$, so $\Phi_\pi$ is feasible. For a given path $\pi = (\mathbf{q}_1, t_1, \ldots, \mathbf{q}_k, t_k)$, the *weight of a path*, denoted $W(\pi)$, is given by $\sum_{i=1}^{k}(t_{i+1} - t_i)c(\mathbf{q}_i, t_i)$.

**Lemma 1.** *For every path $\pi$ in the roadmap, $W(\pi)$ approaches $C(\Phi_\pi)$ as $\Delta t_{\max}$ and $\Delta\mathbf{q}_{\max}$ approach 0 where $\Delta t_{\max}$ denotes the length of the longest time partition and $\Delta\mathbf{q}_{\max}$ denotes the longest distance between adjacent waypoints in the roadmap.*

The proof of this lemma follows fairly simply from the observation that the Lipschitz continuity of $c$ implies that as both $\Delta\mathbf{q}_{\max}$ and $\Delta t_{\max}$ approach 0, the rectangle rule becomes arbitrarily accurate.

**Assumption 2.** *The distribution from which samples are drawn has probability density function $D \in \mathcal{Q} \to \mathbb{R}^+$ which is everywhere nonzero. Furthermore, $D$ is Lipschitz continuous with constant $K_D$.*

This assumption holds for the sampling distribution described in Sec. V-B2 if none of the learned covariance matrices are singular.

**Lemma 2.** *For any error bound $\epsilon > 0$, there exists a choice of $N$ as a function of $\epsilon$, $\Delta t_{\max}$, and $\Delta\mathbf{q}_{\max}$ such that the probability that there exists a path in the roadmap constructed from $N$ waypoints with weight less than $C(\Phi^*)+\epsilon$ approaches 1 as $\Delta t_{\max}$ and $\Delta\mathbf{q}_{\max}$ approach 0.*

The proof of this result revolves around a few key observations. First, the non-zero density of the sampling distribution implies that the probability of sampling arbitrarily close to any configuration along $\Phi^*$ approaches 1. Second, the Lipschitz continuity of $c$ implies that as samples approach the configurations along $\Phi^*$, their costs approach the costs of these configurations. Third, the Lipschitz continuity of $\Phi^*$ implies that as a path with sufficiently small time steps approaches $\Phi^*$, the cost of this path approaches the cost of $\Phi^*$. Finally, as shown in [4] the expansiveness of $\mathcal{Q}_{\text{free}}$ implies that the probability that there exists a path arbitrarily close to $\Phi^*$ in a sufficiently-connected roadmap approaches 1 as the number of samples approaches infinity.

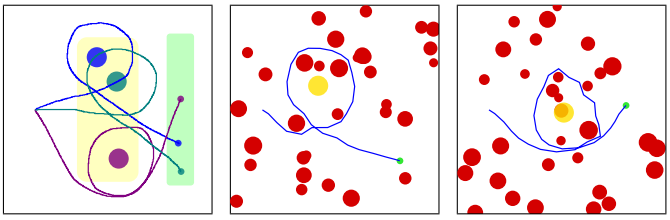


Fig. 6. Left: Three example input demonstrations (blue, violet, teal) are shown as well as the sampling regions for the beacon (yellow) and goal (green). Middle and right: The trajectory (blue) computed by our method for two environments with obstacles (red), a beacon (yellow), and a goal (green).

**Theorem 1.** *As $\Delta\mathbf{q}_{\max} \to 0$ there will exist a path $\pi$ in the roadmap, the piecewise linear interpolation of which has cost $C(\Phi_\pi)$ arbitrarily close to that of the optimal trajectory with probability 1 when $N$ is of order $\Omega(k^{d+2})$ and $\Delta t_{\max} \to 0$ asymptotically faster than $\Delta\mathbf{q}_{\max}$.*

Lemma 2 shows that under these conditions, the weight of the minimum weight path $\pi$ in the roadmap will approach a cost no greater than that of optimal trajectory with probability 1, and Lemma 1 shows that this weight becomes an arbitrarily good approximation of $C(\Phi_\pi)$. Therefore, with probability 1, $C(\Phi_\pi)$ approaches $C(\Phi^*)$.

## VI. Results

We applied DGMP to a simulated 2D point robot and to the Aldebaran NAO small humanoid robot [3]. In the physical experiments, we used 6 joints of the NAO robot: 5 in the right arm and 1 at the hip. Collision detection for motion planning was done using Bullet [37] to detect intersections between a cylindrical approximation of the NAO robot's links and point cloud data obtained from a Microsoft Kinect sensor mounted next to the robot. All computation was performed on a PC with two 6-core 2.0 GHz Intel Xeon E5-2620 processors.

### A. Simulated 2D 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 then stopping at a specified goal. In the learning phase, we used one configuration feature and two landmark features corresponding to the beacon and intended goal position specified by the annotations, each of which had dimension 2. For 6 linearly independent features, the method requires at least 7 demonstrations, which we performed by manually drawing successful trajectories. For the demonstrations, we randomly sampled the beacon location from the yellow region and the goal location from the green region in Fig. 6(left).

In the execution phase, the test environment included 32 circular obstacles which were not present in the demonstrations. We randomly generated 20 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). As can be seen in Fig. 6, the trajectories computed using our method

consistently navigate clockwise around the beacon and reach their intended goal. DGMP was successful in all 20 test cases while the method was never successful when using Euclidean time alignment, indicating that our EM-based approach is important for learning non-trivial tasks.

### B. Physical Task 1: Left-to-Right Powder Transfer Task

In the first physical task, the NAO robot used a spoon to transfer a powder from one container to another in the presence of obstacles as shown in Fig. 7. In our test environment, we placed on a table an instant coffee canister, a cup, and, in some cases, other objects to serve as obstacles. The task was to scoop instant coffee using a spoon and transfer it to the cup without spilling coffee or displacing any objects on the table.

We evaluated DGMP for scenarios in which the coffee canister was always on the right side of the table and the cup was always on the left. This is a simplified version of the general task in which the coffee canister and cup can be anywhere on the table, which will be discussed in Sec. VI-C. We conducted 7 kinesthetic demonstrations for this task, and the locations of the objects on the table were randomized for each demonstration. We drew the positions of the coffee canister and cup from uniform distributions based on 4 inch line segments on the left and right sides, respectively, of the reachable surface of the table.

In the learning phase, we used the configuration motion feature defined by the robot's 6 DOF as well as landmark-based motion features based on the sensed locations of the two task-relevant objects, the canister and cup. For the landmark-based motion features, we considered two points on the robot's end effector (the spoon): the top and bottom surface of the tip of the spoon. (We note that at least two points are required to learn end effector orientations.) We enforced independence between the configuration and each of the landmark-based motion features, as described in Sec. IV-C, so that the largest dependent block in $\mathbf{\Sigma}$ was $6\times6$. Other than the demonstrations, we did not provide the method any input regarding task constraints; e.g., we never explicitly expressed the constraint that the spoon must be level. The learning phase took 2.5 seconds of computation time.

We then created 20 test cases in which the locations of the coffee canister and cup were drawn randomly from the same distribution as the demonstrations. We also placed a bottle on the table as an obstacle at a position drawn uniformly from the reachable surface of the table. The bottle was sufficiently tall that it created a narrow passageway in the NAO's feasible configuration space when the NAO attempted to carry a level spoon over it. A test case was considered *successful* if the robot (1) scooped coffee from the canister and transferred it to the cup without spilling, and (2) did not displace the obstacle, canister, 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 limitations. Of the 20 test cases, 4 were not feasible due to the obstacles being too close to the coffee canister or cup and the robot not having sufficient range of motion. We report statistics for the 16 feasible test cases.

As shown in Fig. 8, the robot running DGMP successfully accomplished the task in 14 of the 16 feasible test cases. The two failures were both due to the Kinect sensor failing to properly sense the extent of the bottle. We also evaluated DGMP using the demonstrations aligned using a simple Euclidean cost metric in configuration space which only considers similarity in joint angles when aligning demonstrations (as in most prior work). Because this Euclidean cost metric does not depend upon the task model, there is no need for EM. Our results show that this approach is ineffective for this task and succeeded in only 9 test cases, indicating that time-alignment that explicitly considers the task model as in the full DGMP approach is beneficial to task success. We also executed the guiding path without motion planning, which resulted in only 8 successful runs due to collision with obstacles.

### C. Physical Task 2: General Powder Transfer Task

We also evaluated DGMP on a more difficult variant of the powder transfer task in which the coffee canister and cup were each permitted to be anywhere on the reachable surface of the table, approximated by a rectangular region spanning 7 inches left to right and 4 inches front to back. This meant that the robot's motions were no longer strictly following a left-to-right trajectory, and thus were more difficult to align. We performed 20 new kinesthetic demonstrations with the coffee canister and cup positions drawn uniformly from the reachable table surface. After completing the demonstrations, the learning phase took 2.6 seconds of computation time. We then created 20 test cases, drawing coffee canister, cup, and bottle obstacle positions randomly from the reachable table surface. Of the 20 test cases, 17 were feasible.

When employing all 20 demonstrations, DGMP succeeded in 16 of the test cases, resulting in a success rate of 94% of the 17 feasible test cases (see Fig. 9). In the one failure case, the obstacle was very close to both the coffee canister and cup, which resulted in a narrow passage in the robot's configuration space that was too narrow for the planner to find a feasible plan in the maximum time allotted (20 seconds). We also evaluated the performance of DGMP for different numbers of demonstrations. When fewer demonstrations are used, the performance of the method degrades gracefully, with a greater than 80% success rate even for just 7 demonstrations.

To illustrate the need for motion planning for this scenario, we also executed the guiding path with no motion planning, resulting in a success rate of under 60%. We also evaluated DGMP using the demonstrations aligned using the simplified Euclidean distance metric and achieved a success rate of 0%. This highlights the benefit of aligning demonstrations by maximizing log-likelihood using our EM-based approach rather than by using the more traditional Euclidean metric, which fails to properly align demonstrations in which the direction of end effector motion varies substantially across demonstrations. A video of a NAO robot performing this task using DGMP is available at: http://robotics.cs.unc.edu/DGMP2.

To illustrate the impact of each component of the DGMP framework, we executed different variants of the motion planner and plot in Fig. 10 the cost of the computed plan based on the learned DGMP metric. Each curve is the average of 6 runs for the same randomly selected test case. As expected,

Fig. 7. Execution of DGMP for the powder transfer task. The robot successfully keeps the spoon level while avoiding obstacles not seen in the demonstrations.

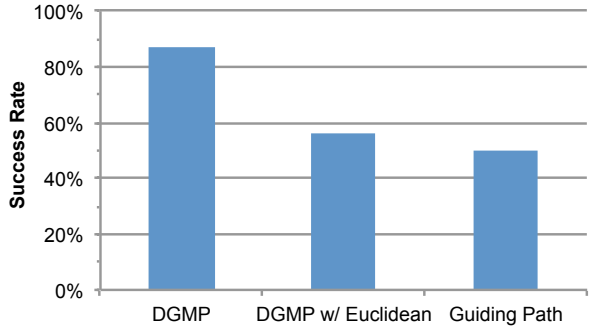



Fig. 8. The performance of DGMP on the left-to-right powder transfer task. We also evaluate the performance of DGMP using Euclidean time alignment (rather than maximizing learned likelihood) and also executing the guiding path (without sampling-based motion planning).

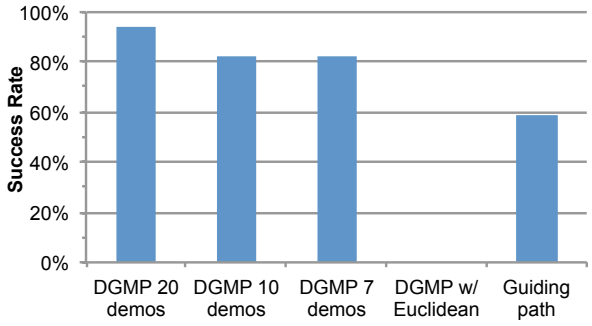



Fig. 9. The performance of DGMP on the general powder transfer task when 20, 10, and 7 demonstrations are provided to the learning phase. We also evaluated the performance of DGMP using Euclidean time alignment (rather than maximizing learned likelihood) and the guiding path (without sampling-based motion planning). DGMP performed better with more demonstrations, but still exceeded an 80% success rate when only 7 demonstrations were provided.

allowing more computation time results in lower cost plans. As described in Sec. V, the DGMP cost metric can be used with either PRM or RRG against which we will compare our proposed extensions that accelerate performance by biasing sampling based on the learned metric, seeding along the guiding path, and incorporating layers. DGPRM and DGRRG, which both include all the speedups, are roughly equivalent for this application. We also show DGPRM with some of its components removed (i.e., removing speedups gained by layers and/or seeding). The results show that the biggest speedup in DGPRM comes from biasing configuration samples based on the learned cost metric during roadmap expansion. DGPRM and DGRRG are both over 20 times faster than the traditional PRM algorithm for plans of equivalent cost.

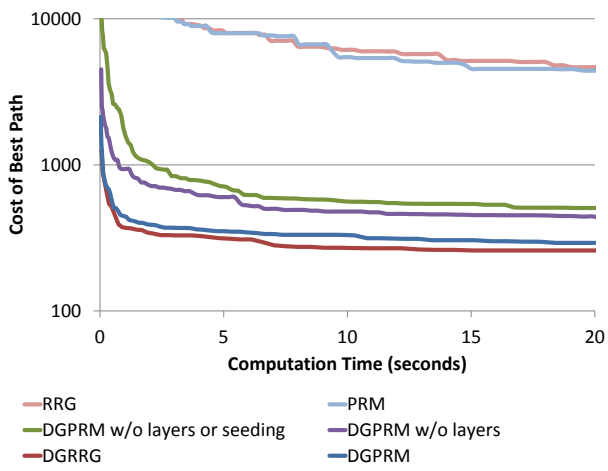



Fig. 10. The cost of the best solution found as a function of roadmap planning time for different variants of the method applied to the general powder transfer task. Note the logarithmic scale on the vertical axis.

### D. Physical Task 3: Push a Button

We also considered the task of pressing a button, where the button may be positioned on a table, a slanted surface, or even a vertical wall. To correctly perform the task, the robot needed to learn how to push a button in any of these orientations from the same set of demonstrations. Furthermore, additional obstacles were introduced into the execution environment.

To train the method, we performed 9 demonstrations of pressing a 3 cm diameter button. In 3 of these demonstrations, the button was placed randomly on the reachable surface of a table; in another 3, the button was randomly placed on the reachable surface of a plane inclined 40 degrees; and in the final 3, the button was randomly affixed to the reachable surface of a vertical wall in front of the robot.

The motion features we used were the 6 joint angles of the robot's right arm and hip and the 3-dimensional positions of both the hand and finger relative to the pose of the button. As before, we enforced independence in the covariance matrix between the configuration motion feature and each of the two landmark-based motion features. The learning phase took 0.9 seconds of computation time.

To test the method, we considered 4 scenarios, and performed 3 random tests on each for a total of 12 test cases. The scenarios, shown in Fig. 11, included (1) placing the button and a non-convex obstacle randomly on the reachable surface

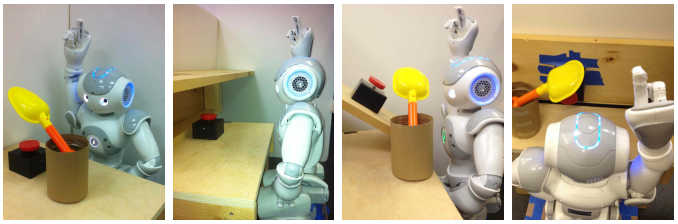


Fig. 11. Scenarios for the button pushing task.

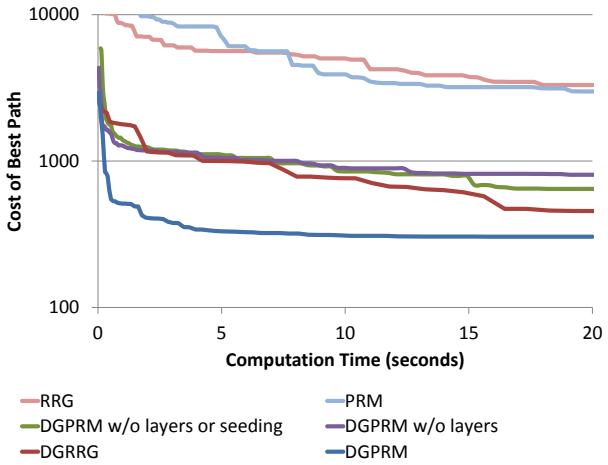



Fig. 12. The cost of the best solution found as a function of roadmap planning time for different variants of the method applied to the button pushing task.

of the table, (2) placing a shelf over the table and placing the button randomly on the table under the shelf, (3) placing the button randomly on a plane inclined at 40 degrees and placing a non-convex obstacle randomly beside the inclined plane such that it hung over the inclined plane, and (4) placing the button on a vertical wall and placing a tall obstacle randomly on the surface of the table.

In Fig. 12 we show the rapid convergence of DGPRM compared to regular RRG and PRM without layers or accelerations based on the learned task model. While DGRRG did find lower cost paths for the same number of samples, we see in the figure that DGPRM performed better because each sample could be generated more quickly due to greater opportunities for parallel execution. We believe this is because the biased sampling distribution derived from the demonstrations largely subsumes the role of RRG's roadmap expansion approach in effectively biasing samples towards the relevant portions of the configuration space.

Fig. 13 shows the execution of a DGMP plan. We considered an execution successful if it avoided obstacles and depressed the button. DGMP succeeded in 11 of the 12 test cases, a greater than 90% success rate. The sampling-based motion planner was crucial to success in this task as the guiding path was successful in only 1 of the 12 tests cases.

## VII. Conclusion and Future Work

We presented demonstration-guided motion planning (DGMP), a new framework for planning motions for assistive robots to perform tasks in unstructured environments such as homes or offices. DGMP combines the strengths

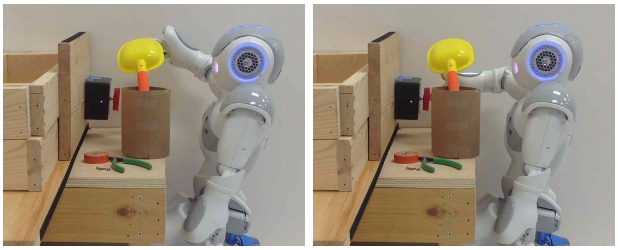


Fig. 13. Execution of a DGMP plan for one of the button pushing scenarios.

of demonstration-based learning and sampling-based motion planning to generate motion plans that (1) aim to satisfy learned features of the motion that are required for the task to be successfully accomplished and (2) avoid obstacles in unstructured environments. We use kinesthetic demonstrations and statistical modeling methods to learn a time-dependent cost metric that encodes features of a task's motion that are consistent across the demonstrations and, hence, are likely required to successfully execute the task. We formalize the cost metric as a Mahalanobis distance between a planned trajectory and the distribution of demonstrations in a feature space parameterized by the locations of task-relevant objects. Our new asymptotically-optimal sampling-based motion planner computes plans that simultaneously avoid obstacles and asymptotically globally minimize the learned cost metric. The planner also leverages the demonstrations to significantly reduce motion plan computation time. We showed the effectiveness of combining learning with sampling-based motion planning on the NAO robot performing assistive tasks.

In future work, we plan to extend DGMP to work effectively for a broader class of problems. The method currently is designed for static execution environments; we plan to extend it to dynamic environments with moving obstacles, which will require real-time perception. Extensions for partial and noisy sensing would be beneficial. Also, the method currently is designed for tasks which can be modeled as Gaussians in the motion feature space. We plan in future work to consider automatic identification of relevant motion features along with non-Gaussian distributions, which will impact both the learning and execution phases. We will also investigate integrating our approach with task-level planning.

## References

[1] 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.

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

[3] Aldebaran Robotics, "Aldebaran Robotics NAO for education," http://www.aldebaran-robotics.com/en/naoeducation, 2010.

[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.

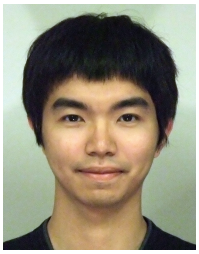
[5] L. Jaillet, J. Cortes, and T. Simeon, "Sampling-based path planning on configuration-space costmaps," *IEEE Trans. Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

[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] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. Int. Conf. Machine Learning (ICML)*. New York, New York, USA: ACM Press, 2004.
[19] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, 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.
[20] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Int. Conf. Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 182–189.
[21] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, no. 3, 2011, pp. 1561–1566.
[22] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sep. 2008, pp. 1083–1090.
[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] S. J. Lee and Z. Popović, "Learning behavior styles with inverse reinforcement learning," in *ACM Trans. Graphics (SIGGRAPH 2010)*, vol. 29, no. 4, Jul. 2010, pp. 122:1–122:7.
[25] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, Jun. 2009.
[26] 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.
[27] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard, "Imitation learning with generalized task description," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 3968–3974.
[28] 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.
[29] S. Calinon, *Robot Programming by Demonstration*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.
[30] K. Bernardin, K. Ogawara, K. Ikeuchi, and R. Dillmann, "A Sensor Fusion Approach for Recognizing Continuous Human Grasping Sequences," *IEEE Trans. Robotics*, vol. 21, no. 1, pp. 47–57, 2005.
[31] N. T. Nguyen, D. Q. Phung, and S. Venkatesh, "Learning and Detecting Activities from Movement Trajectories Using the Hierarchical Hidden Markov Model," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 955–960, 2005.
[32] 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.
[33] 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.
[34] P. C. Mahalanobis, "On the generalised distance in statistics," *National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.
[35] 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.
[36] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 431-441, 1963.
[37] "Bullet Physics Library," 2012. [Online]. Available: http://bulletphysics.org

**Chris Bowen** received the B.S. degree from the Georgia Institute of Technology, Atlanta, GA, USA, in 2011 and the M.S. degree from the University of North Carolina at Chapel Hill, NC, USA, in 2013, both in computer science. He is currently pursuing a Ph.D. in computer science at the University of North Carolina at Chapel Hill.

His current research focuses on applications of machine learning to robotic systems.



**Gu Ye** received the B.S. degree from Zhejiang University, China and the M.S. degree from the University of North Carolina at Chapel Hill, NC, USA, in 2011. His research spanned robot learning and motion planning.

**Ron Alterovitz** received his B.S. degree with Honors from Caltech, Pasadena, CA in 2001 and the Ph.D. degree in industrial engineering and operations research at the University of California, Berkeley, CA, USA in 2006.

In 2009, he joined the faculty of the Department of Computer Science at the University of North Carolina at Chapel Hill, NC, where he leads the Computational Robotics Research Group. His research focuses on motion planning for medical and assistive robots. Prof. Alterovitz has co-authored a book on Motion Planning in Medicine, was awarded a patent for a medical device, has received multiple best paper finalist awards at IEEE robotics conferences, and is the recipient of the NIH Ruth L. Kirschstein National Research Service Award and the NSF CAREER Award.