# Learning Cost Function and Trajectory for Robotic Writing Motion

Hang Yin[1,2], Ana Paiva[2] and Aude Billard[1]

***Abstract*— We present algorithms for inferring the cost function and reference trajectory from human demonstrations of hand-writing tasks. These two key elements are then used, through optimal control, to generate an impedance-based controller for a robotic hand . The key novelty lies in the flexibility of the feature design in the composition of the cost function, in contrast to the traditional approaches that consider linearly combined features. Cross-entropy-based methods form the core of our learning technique, resulting in sample-based stochastic algorithms for task encoding and decoding. The algorithms are validated using an anthropomorphic robot hand. We assess that the correct compliance is well encapsulated by subjecting the robot to perturbations during task reproduction.**

***Index Terms*— learning from demonstrations, stochastic optimization, impedance control**

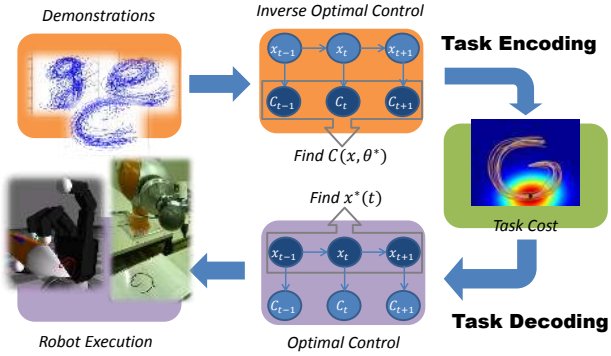



Fig. 1. Learning robotic writing through encoding and decoding task cost

## I. INTRODUCTION

The efficient acquisition of skills is of general interest to robotics, and in particular to high degrees of freedom (DOFs) anthropomorphic robots, for which complex programming is required. By enabling human experts to teach robots via demonstrating the task intuitively, learning from demonstrations, also called imitation learning, offers a fundamental framework for addressing this challenge. To robot agents, it is essential to develop approaches for encoding and decoding the demonstrated skills. Much research work proposes to represent and execute task policies in a direct way: by exploiting supervised learning techniques, the policy is encoded by a regressor or dynamical system [1][2][3] trained on demonstrated state and action dataset. Then the robot executes the policy by following the learned model in a straightforward way.

In contrast to direct approaches, another way is to implicitly represent and derive a policy with a cost function. And the demonstrated behavior is assumed to be optimal or suboptimal in terms of the cost function. This is formulated as an inverse optimal control problem, that provides a more succinct representation of the underlying task and a possibility to derive a policy for the robot with distinct embodiments in novel task scenarios. Most inverse optimal control [4] research assumes that the cost function to learn is linear with unknown parameters. Also, gradient-based methods are widely used [5][6] to solve the resulting optimization problem. In this paper, we consider the problem of extracting a tracking trajectory as well as the deviance penalty defined in the local frame of reference, which leads to a nonlinear parameterization. Gradient-based methods are ill-suited for solving such non-convex optimization problem as they tend to end up with poor local optima. Also, derivation of gradients to explore the feature design for each model is error-prone and not applicable for model-free problems. Thus there is a desire to develop other approaches to deal with these challenges.

With a learned cost function, we can derive a policy through optimal control or reinforcement learning. We cast this as the decoding of a learned task. Plenty of policy search and trajectory optimization approaches have been proposed. Among these approaches, sampling-based stochastic-optimization methods are gaining momentum [7][8][15]. The advantage of sampling-based methods lies in their strength in model-free learning and fast convergence to good solutions. We base our trajectory optimization on a similar concept, but in the context of a high-dimensional multi-manipulator system.

In this paper, we propose to use the cross-entropy method, a stochastic optimization algorithm to learn the cost function with the parameterization that encodes trajectory tracking, based upon the maximum-entropy principle. The method is also used to derive an optimal motion trajectory for learned costs on a multi-manipulator system. The method relies the evaluation of samples without knowing the explicit model, which demonstrates the potential of the algorithm as a model-free approach. Furthermore, from a model-based perspective, we show how the proposed cost parameterization encapsulates demonstrated behavior in terms of motion compliance. Figure 1 illustrates overall flow of our approach. The main contributions of this paper are highlighted as below.

- A cross-entropy-like method is developed to deal with the challenge of learning cost function with a nonlinear

[1]Hang Yin and Aude Billard are with the Learning Algorithms and Systems Laboratory (LASA), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Email: {hang.yin, aude.billard}@epfl.ch

[2]Hang Yin and Ana Paiva are with the Intelligent Agents and Synthetic Characters Group (GAIPS), INESC-ID and Instituto Superior Técnico, University of Lisbon, Portugal. Email: ana.paiva@inesc-id.pt

parameterization form, as in this case that the features are not linear independent;
- The sampling schema of original algorithms is extended to allow sampling in the nullspace of parameters with a feature representation. This extension can be used to embed features and prior knowledge to facilitate trajectory optimization in the phase of task decoding.

Apart from the work reviewed above, [9] also proposes an intrinsic cost with a similar quadratic parameterization. And, a stochastic method is employed to learn the parameters. The difference lies in that our weight matrix is defined in the local reference frame of the tracking trajectory. Moreover, the policy derivation in [9] is realized with AICO [10], whereas we use a trajectory sampling method. AICO relies on the duality between optimal estimation and control. It performs probabilistic inference with extended Kalman smoothing, thus a local linearization is desirable for Gaussian message propagation. In our work, the cross-entropy-based method requires only forward trajectory evaluations and can hence use non-differentiable objectivities or constraints without explicit model knowledge. Finally and most importantly, [9] requires access to an extrinsic task cost as a critic to the internal planning system. Our approach, which is situated in the context of imitation learning, holds no assumption of the extrinsic cost but requires demonstrations from an expert.

## II. PROBLEM DEFINITION

We consider the problem of transferring skills to a robot with given demonstrated trajectories $\{\boldsymbol{x}_t^*\}$, where $\{\boldsymbol{x}_t^*\}$ denotes the state trajectory of interest. Following implicit learning from demonstrations, it is assumed that $\{\boldsymbol{x}_t^*\}$ is optimal or suboptimal with respect to an unknown cost $C(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the parameters to learn. Note that the time parameter $t$ is omitted for the brevity of notations. In order to mimic the demonstrator, the robot is required to derive its own favorable behavior $\{\boldsymbol{q}_t^*\}$ by minimizing the sum of $C(\boldsymbol{x}, \boldsymbol{\theta})$ along the optimal state trajectory.

The problem can be divided into two phases. The first part which aims to reveal unknown costs can be formulated as an inverse optimal control problem. In general, this problem is ill-posed as there are ambiguous results (e.g., constant cost) that always fulfill the optimality of demonstrations. One elegant way to address this is with the maximum-entropy framework (MaxEnt) [5], where trajectories are assumed to be subject to a Boltzmann distribution. By exploiting this concept, we can estimate cost parameters by maximizing the likelihood of demonstrations under this distribution:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(\tau^*|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \frac{\exp(-J(\tau^*, \boldsymbol{\theta}))}{\int_\tau \exp(-J(\tau, \boldsymbol{\theta}))} \quad (1)$$

where $\tau^* = \{\boldsymbol{x}_{1:T}^*\}$ and $\tau = \{\boldsymbol{x}_{1:T}\}$ denote demonstrated and all possible trajectories with a time horizon of $T$, respectively. $J(\tau) = \sum_{t=1}^{T} C(\boldsymbol{x}_t, \boldsymbol{\theta})$ defines the accumulated cost along trajectory $\tau$.

The second part of the problem is to derive robot optimal trajectory $\tau_q^* = \{\boldsymbol{q}_{1:T}^*\}$ given the established cost. This is a typical optimal control problem, and we formulate it as finite horizon trajectory optimization as follows:

$$\tau_q^* = \underset{\tau_q}{\operatorname{argmin}}\, J(\tau_q) = \underset{\tau_q = \{\boldsymbol{q}_{1:T}\}}{\operatorname{argmin}} \sum_{t=1}^{T} C(\kappa(\boldsymbol{q}_t), \boldsymbol{\theta}) \quad (2)$$

where $\tau_q$ denotes trajectory applied on robot and $\kappa$ is a kinematic function. Some remarks are given for the problems formulated in (1) and (2):

- The state trajectory and system can also be indexed with a phase variable $z$ for the generality of the model.
- Dynamics can be introduced for both (1) and (2). They can be either known as $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ or learned from set $\{\boldsymbol{x}_t, \boldsymbol{u}_t\}$, where $\{\boldsymbol{u}_t\}$ denotes the control to the dynamics.
- Robot state trajectory $\{\boldsymbol{q}_t\}$ does not necessarily appear as direct features in the task cost. For instance, the robot trajectory might be featured in joint space but the feature of cost might be the trajectory of the end-effector or manipulated objects. We assume there is a mapping function $\kappa$ (not necessarily known to the algorithm) to convert $\{\boldsymbol{q}_t\}$ to $\{\boldsymbol{x}_t\}$.

## III. PROPOSED METHODS

In this section, we present approaches for addressing the problems formulated above. We first give a detailed parameterization of the cost function and highlight its difference and challenges compared with other work. Then we introduce the general cross-entropy method as a core technique for dealing with these challenges. We also propose to sample in the nullspace of parameters in the cross-entropy method. Finally, the algorithms for encoding and decoding tasks, as well as the development of compliance behavior, are discussed and listed.

### A. Parameterization of Cost Function

The cost function defined in (1) and (2) is of a general form. The concrete parameterization of $C(\boldsymbol{\theta})$ determines the parameters to infer and the features captured to encode the underlying task. Much work proposes to use a form that consists of linear combination of features. This is advantageous as (1) turns out to be a convex problem. Here, for the purpose of trajectory tracking, we propose a different form with an unknown trajectory and weight matrices as parameters. This can be formulated as a quadratic form similar to [9] such as

$$C(\boldsymbol{x}, \boldsymbol{\theta}) = (\boldsymbol{x} - \boldsymbol{x}_t^{ref})^T \boldsymbol{Q}_{t,\{\boldsymbol{x}_t^{ref}\}} (\boldsymbol{x} - \boldsymbol{x}_t^{ref}) \quad (3)$$

where $\boldsymbol{\theta} = \{\boldsymbol{x}_t^{ref}, \boldsymbol{Q}_{t,\{\boldsymbol{x}_t^{ref}\}}\}$. $\{\boldsymbol{x}_t^{ref}\}$ denotes a state trajectory (e.g., letter calligraphy in a 2D case) to track and $\{\boldsymbol{Q}_{t,\{\boldsymbol{x}_t^{ref}\}}\}$ is a trajectory of positive definite matrices that possibly *depend* on $\{\boldsymbol{x}_t^{ref}\}$. In the following sections, we use $\boldsymbol{Q}_t$ to denote $\boldsymbol{Q}_{t,\{\boldsymbol{x}_t^{ref}\}}$ for brevity.

In [9], $\boldsymbol{Q}_t$ is diagonal. This implies the error of reference tracking will be independently penalized by the diagonal weights along axes of a *fixed global* reference frame. In contrast to such form, we propose to define $\boldsymbol{Q}_t$ in a *local* frame with respect to the reference trajectory. This enables us

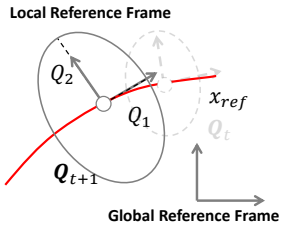



Fig. 2. Parameter definition under global and local reference frames

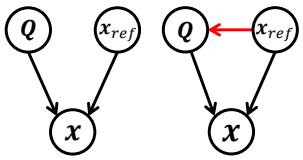



Fig. 3. Graphical models for dependency among variables: left - $\boldsymbol{Q}_t$ independent from unknown $\boldsymbol{x}_t^{ref}$, as in [9]; right - $\boldsymbol{Q}_t$ dependent on $\boldsymbol{x}_t^{ref}$

to weight the errors in a moving reference frame that captures local geometrical features (e.g., normal direction of reference trajectory). From the perspective of the global frame, this weight matrix $\boldsymbol{Q}_t$ can be illustrated by a hyper-ellipsoid with varying principle axes length, as well as orientation in alignment with the reference trajectory (See Figure 2).

The difference on the definition of $\boldsymbol{Q}_t$ between [9] and our work can be further demonstrated in Figure 3. In [9], there is no prior assumption about the dependency between $\boldsymbol{Q}_t$ and $\boldsymbol{x}_t^{ref}$. The resulting model is comparatively sparse such that the inference of unknown parameters might be straightforward (e.g., using mean trajectory as the reference under the Gaussian assumption). While in our case (right graph in Figure 3), defining $\boldsymbol{Q}_t$ with respect to $\boldsymbol{x}_t^{ref}$ introduces a prior dependency. Such dependency implies a constraint between the parameters to infer. The parameterization of the resulting cost function is a nonlinear combination of these unknown variables.

### B. Cross Entropy Method

The cost function (3) is of a parameterization with nonlinear composition. Gradient derivation for the resulting problem is nontrivial and requires re-parameterization for a specific model. This motivates us to exploit cross-entropy (CE) method, a stochastic approach, to infer the unknown parameters.

The CE method considers a general optimization $l = \min J(\boldsymbol{x})$ as a sequence of rare-event probability estimation problem, by seeking $\{l_i\}$ and $\{q_i\}$ to evaluate

$$\gamma_i = E_{q_i}(I_{\{J(\boldsymbol{x})<l_i\}}) = E_{q_{i+1}}(I_{\{J(\boldsymbol{x})<l_i\}}\frac{q_i(\boldsymbol{x})}{q_{i+1}(\boldsymbol{x})}) \quad (4)$$

where $I_{\{\cdot\}}$ is an indicator function and $\{q_i\}$ is assumed to belong to a family of distributions as proposals. An optimal importance sampler $q_{i+1}$ can be found by solving an empirical form

$$\hat{q}_{i+1} = \underset{q}{\operatorname{argmax}} \frac{1}{N}\sum_{j=1}^{N} I_{\{J(\boldsymbol{x}_j)<l_i\}} \ln(q(\boldsymbol{x}_j)) \quad (5)$$

where $\{\boldsymbol{x}_j\}$ are $N$ instances sampled from $q_i$. We give a brief description about the iteration procedure of CE method and some remarks related to our application below. For more detailed derivation, interested readers can refer to [11].

i. With an initial density $q$ as sampling distribution, generate a set of samples $\{\boldsymbol{x}_j\}$, $j = 1, ..., N$;

ii. Assign weights to sampled instances to construct an elite set, e.g., define the membership of the set by evaluating $J(\boldsymbol{x}_j) < l_i$ where $l_i$ can be $(1-\rho)$-quantile of evaluated performance;

iii. Estimate density $\hat{q}$ through (5) and use $\hat{q}$ as the new sampling distribution;

iiii. Iterate steps i. through iii. until stop condition is fulfilled;

We choose to use a multivariate Gaussian as the sampling distribution $q$, as it yields a closed form solution for (5). Also, a soft version of membership function $I_{\{\cdot\}}$ is used. The standard CE method, as in ii., uses a hard threshold to classify samples (either elite or not), and then elite samples are indiscriminately treated in the estimation of the new sampling distribution. Other variants consider assigning importance to each sample according to their performance evaluation. For instance, in Covariance Matrix Adaptation (CMA-ES), it is suggested to use weight that is proportional to the inverse of performance within the elite set. We adopt a membership function similar to [8], that all samples are taken into account by weighing the normalized exponential values of their relative performance.

Although the CE method globally explores the state space of $\boldsymbol{x}$, its global optimality is guaranteed in a probabilistic manner. In practice, the routine will converge to a local solution if no sample is generated in the vicinity of the global optimum.

### C. Feature Representation and Nullspace Sampling

The CE method requires sampling in the parameter space to explore solution. For high-dimension space such as trajectory, it might be more efficient to sample in the feature space that is rich enough for sampling good solutions. In order to encode the reference trajectory and varying diagonals of $\boldsymbol{Q}_t$ in (1), as well as the optimized trajectory in (2), we propose to use a function approximator to represent trajectories and to sample in the corresponding feature space. For instance, a trajectory can be approximated with a linear combination of $M$ normalized Radial Basis Function (RBF) features

$$\boldsymbol{x}_{ref}(t) = \boldsymbol{\omega}^T\boldsymbol{\Phi}(t) = \sum_{i=1}^{M}\boldsymbol{\omega}_i\frac{\exp(-\alpha(t-t_i)^2)}{\sum_{j=1}^{M}\exp(-\alpha(t-t_i)^2)} \quad (6)$$

where $t$ can also be replaced with a phase variable $z$ to have a general representation.

Sometimes we might expect sampled trajectories to fulfill some constraints, e.g., to pass through a specific point. This is especially useful in trajectory optimization when we expect to have all the samples start from initial state $x_0$ or fix both boundary points. We propose to address this by sampling in the nullspace of the feature parameter space. Concretely, suppose $\boldsymbol{\omega}$ is required to generate trajectories constrained on $\boldsymbol{x}_{ref}^{const}$

$$\boldsymbol{\omega}^T[\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_c] = \boldsymbol{x}_{ref}^{const} = [\boldsymbol{x}_1^{const}, ..., \boldsymbol{x}_c^{const}] \quad (7)$$

We can find a linear transformation matrix $\boldsymbol{R}$ through Singular Value Decomposition (SVD) to ensure

$$(\boldsymbol{\omega} + \boldsymbol{R}\delta\boldsymbol{\omega})^T[\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_c] = [\boldsymbol{x}_1^{const}, ..., \boldsymbol{x}_c^{const}] \quad (8)$$

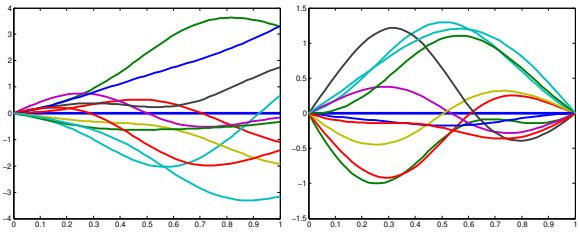


Fig. 4. Nullspace trajectory sampling: fixing starting and boundary points

holds for any $\delta\boldsymbol{\omega}$ sampled in the subspace of the feature parameter space. Sampling in this nullspace has two advantages. It is possible to efficiently explore trajectories without needing to reject those that violate the constraints. Figure 4 shows sampled trajectories with fixed end points. Also, because $\boldsymbol{R}$ is a linear transformation, perturbed parameter $\boldsymbol{\omega} + \boldsymbol{R}\delta\boldsymbol{\omega}$ is still subject to a normal distribution, given $\delta\boldsymbol{\omega}$ is sampled as Gaussian noise.

### D. Algorithms

*1) Learning Cost Function for Task Encoding:* To learn the task cost by solving (1), we can employ the CE method and feature sampling presented above. A typical challenge to solving inverse optimal control problem as (1) is to evaluate the denominator. This is indeed to calculate the partition function of a Boltzmann distribution, and it is related to solving an optimal control problem. We estimate this term with $K$ locally sampled trajectories from a proposal distribution $\gamma$ (e.g., a Gaussian centered at the optimal solution)[1]. Also, we rewrite (1) as minimizing the negative log likelihood, thus (1) is converted to

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^{D} \log \frac{\exp(-J(\tau^*, \boldsymbol{\theta}))}{\sum_{k=1}^{K} \frac{1}{\gamma(\hat{\tau}_k)} \exp(-J(\hat{\tau}_k, \boldsymbol{\theta}))} \quad (9)$$

where $\hat{\tau}_k = \{\hat{\boldsymbol{x}}_{1:T}^k\}$ is the locally sampled trajectory, $\boldsymbol{\theta} = \{\boldsymbol{x}_t^{ref}, \boldsymbol{Q}_t\}$ are the learning parameters and $D$ denotes the number of demonstrations. With an initial guess of parameters and its distribution $p$, we can iterate Algorithm 1 to find parameters that encode task costs. Here, Gaussian sampling is used, thus the distribution can be denoted as $p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We present some remarks about the algorithm arguments and implementation in practice:

- $\boldsymbol{\theta}$ and $p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be initialized with some uninformative values, such as a straight line for the reference trajectory.
- Larger number of samples for the CE method and the partition function evaluation lead to a better estimation, if more computational budget is available for each iteration step.
- The update of distribution parameters can be smoothed by introducing a proportional factor as suggested in [11], which leads to more stable iterations in general.

[1]For a Gaussian distribution, a closed-form solution can be calculated in this case. A sample-based evaluation is used here for the generality of the algorithm.

**Algorithm 1 Encoding** - Iteration for Learning Cost Function based on Cross Entropy Stochastic Optimization

**Require:** $\boldsymbol{\theta} = \{\boldsymbol{x}_t^{ref}, \boldsymbol{Q}_t\}, p(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}), \gamma, C(\boldsymbol{x}, \boldsymbol{\theta}), K, N$ - Number of samples, $\boldsymbol{D}$ - Demonstrations of $T$ length
**Ensure:** ${\boldsymbol{\theta}}^{New}$, $p({\boldsymbol{\mu}_{\boldsymbol{\theta}}}^{New}, {\boldsymbol{\Sigma}_{\boldsymbol{\theta}}}^{New})$
  **for all** $i$ in 1:$N$ **do**
    $\hat{\boldsymbol{\theta}}_i \leftarrow p(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ ▷ Sample parameters according to current distribution. Apply projection from nullspace if necessary
    **for all** $j$ in 1:$K$ **do**
      $\hat{\tau}_j = \{x_t^k, t = 1, ..., T\} \leftarrow \gamma$ ▷ Sample locally perturbed trajectories for evaluating partition function, see Figure 5
    **end for**
    $L_i \leftarrow -\sum_{i=1}^{D} \log \frac{\exp(-J(\tau^*, \boldsymbol{\theta}))}{\sum_{k=1}^{K} \frac{1}{\gamma(\hat{\tau}_k)} \exp(-J(\hat{\tau}_k, \boldsymbol{\theta}))}$
  **end for**
  $\{\hat{\theta}_j\}_{elite} \leftarrow \text{EliteSet}(\{\hat{\boldsymbol{\theta}}_i, L_i\})$ ▷ Construct elite set
  ${\boldsymbol{\theta}}^{New}, {\boldsymbol{\mu}_{\boldsymbol{\theta}}}^{New} \leftarrow \text{Mean}(\{\hat{\boldsymbol{\theta}}_j\}_{elite})$
  ${\boldsymbol{\Sigma}_{\boldsymbol{\theta}}}^{New} \leftarrow \text{Covar}(\{\hat{\boldsymbol{\theta}}_j\}_{elite})$ ▷ Update parameters

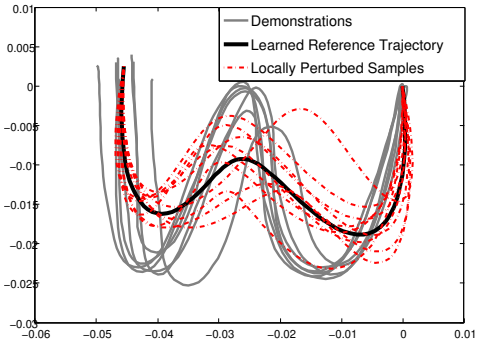



Fig. 5. Sampling perturbed trajectories for approximation of partition function evaluation

*2) Generating Motion Trajectory as Task Decoding:* Given the learned cost function, we can derive motion trajectories for a robot, which can be considered to decode the task. Taking the view of [12], this is equivalent to infer a latent state trajectory on a factored graph. The trajectory is featured in joint space as $\{\boldsymbol{q}_t\}$, and can be converted to the state space of the cost function through $\kappa(\boldsymbol{q}_t)$. Note that the complexity of $\kappa$ depends on the robot embodiment as well as the task environment. It can be a kinematic function for the free motion of a single manipulator, or other nontrivial forms, e.g., consider $\kappa$ that correlates the joint trajectory of a anthropomorphic hand to the motion of a manipulated object. Sample-based inference, such as the CE method, can approach the problem without an explicit access to $\kappa$. Thus it works as a model-free method. A single iteration step is given as Algorithm 2. The trajectory is optimized by searching $\boldsymbol{\beta}$, which parameterizes a function approximator with feature $\boldsymbol{\Phi}$ for each DOF. Here we denote the parameters and features with notations different from (6), as the algorithm is also open to parameterize the trajectory with other features.

**Algorithm 2 Decoding** - Iteration for Deriving Trajectory based on Cross Entropy Stochastic Optimization

**Require:** $\boldsymbol{\beta}, \boldsymbol{\Psi}, p(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}), C(\kappa(\boldsymbol{q}_t)), N$ - Number of samples

**Ensure:** ${\boldsymbol{\beta}}^{New}$, $p({\boldsymbol{\mu}_{\boldsymbol{\beta}}}^{New}, {\boldsymbol{\Sigma}_{\boldsymbol{\beta}}}^{New})$

**for all** $i$ in 1:$N$ **do**

$\hat{\boldsymbol{\beta}}_j \leftarrow p(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}})$ ▷ Sample trajectory parameters

$\{\boldsymbol{q}_t\}_i \leftarrow \hat{\boldsymbol{\beta}}_i^T \boldsymbol{\Psi}$ ▷ Evaluate robot DOF trajectory.

Apply projection from nullspace if necessary

$J(i) \leftarrow \sum_{t=1}^{T} C(\kappa(\boldsymbol{q}_t))$

**end for**

$\{\hat{\beta}_j\}_{elite} \leftarrow EliteSet(\{\hat{\boldsymbol{\beta}}_i, J_i\})$ ▷ Construct elite set

${\boldsymbol{\beta}}^{New}, {\boldsymbol{\mu}_{\boldsymbol{\beta}}}^{New} \leftarrow$ Mean$(\{\hat{\boldsymbol{\beta}}_j\}_{elite})$

${\boldsymbol{\Sigma}_{\boldsymbol{\beta}}}^{New} \leftarrow$ Covar$(\{\hat{\boldsymbol{\beta}}_j\}_{elite})$ ▷ Update parameters

*3) Deriving Impedance Controller through Model-based Optimal Control:* The varying weight matrix $\boldsymbol{Q}_t$ encodes the necessity of rejecting disturbance in specific directions during the motion. This encapsulated feature can be straightforwardly used to derive the compliant behavior of a robot in the framework of model-based optimal control. Suppose the motion of tooltip can be modeled as a discrete-time linear stochastic dynamical system, such as

$$\bar{\boldsymbol{x}}_{t+1} = \boldsymbol{A}_t \bar{\boldsymbol{x}}_t + \boldsymbol{B}_t \boldsymbol{u}_t + \boldsymbol{\nu}_t \tag{10}$$

where $\bar{\boldsymbol{x}}_t = (\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t)^T$ is the augmented state vector that incorporates velocity, and $\boldsymbol{\nu}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ models the white noise of the system. $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are system parameters and $\boldsymbol{u}_t$ denotes the input signal to the system. The time index is retained as it can also be used to represent the linearization of a general nonlinear system.

The learned cost $(\boldsymbol{x} - \boldsymbol{x}_{ref})^T \boldsymbol{Q}_t (\boldsymbol{x} - \boldsymbol{x}_{ref})$ is applied by adding a term that penalizes large input as

$$\bar{C}(\bar{\boldsymbol{x}}_t, \boldsymbol{u}_t, t) = (\bar{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_{ref})^T \bar{\boldsymbol{Q}}_t (\bar{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_{ref}) + \boldsymbol{u}_t^T \boldsymbol{R} \boldsymbol{u}_t \tag{11}$$

where $\bar{\boldsymbol{Q}}_t$ is an augmented matrix with $\boldsymbol{Q}_t$ and small values (e.g., $10^{-4}$ in our experiment) as diagonals if only $\boldsymbol{x}_{ref}$ needs to be tracked.

Solving $\{\boldsymbol{u}_t\}$ with respect to (10) and (11) can be cast as a classic Linear-Quadratic-Gaussian (LQG) optimal control or inference on a Dynamical Bayesian Network, where both system transition and emission distributions are linear Gaussian. By exploiting the linearity and quadratic form of learned cost, an exact solution can be obtained as

$$\boldsymbol{u}_t^* = \boldsymbol{L}_t (\bar{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_{ref}) \tag{12}$$

where $\boldsymbol{L}_t$ is the feedback gain that can be recursively evaluated by following Riccati equation

$$\boldsymbol{S}_t = \bar{\boldsymbol{Q}}_t + \boldsymbol{A}_t^T (\boldsymbol{B}_t \boldsymbol{R}^{-1} \boldsymbol{B}_t^T + \boldsymbol{S}_{t+1}^{-1})^{-1} \boldsymbol{A}_t \tag{13}$$

$$\boldsymbol{L}_t = -\boldsymbol{R}^{-1} \boldsymbol{B}_t^T (\boldsymbol{B}_t \boldsymbol{R}^{-1} \boldsymbol{B}_t^T + \boldsymbol{S}_{t+1}^{-1})^{-1} \tag{14}$$

where $\boldsymbol{S}_T = \bar{\boldsymbol{Q}}_T$ for a problem terminates at $T$.

The control input is linked to a tracking error and a varying gain in (12). This can be considered as an impedance controller, with $\boldsymbol{L}_t$ as the impedance parameter. It is observed from (13) and (14) that the impedance co-varies with $\bar{\boldsymbol{Q}}_t$, thus it implies that the robot needs to be stiff when $\boldsymbol{Q}_t$ is large, and vice versa.

## IV. EXPERIMENTS

### A. A Simple Example

We first validate our algorithm for learning cost function on a simple synthetic example. In this experiment, the motion of a particle is simulated as a second-order dynamical system and the particle is expected to track a straight horizontal line. As the particle proceeds, perturbations are applied within a small time window along vertical direction. The amplitude of perturbations is modeled as Gaussian noise, where the variance varies in different sections. As all resultant trajectories are taken as good demonstrations, it is expected to find a cost function that can capture the information of varying perturbations in weight matrices.

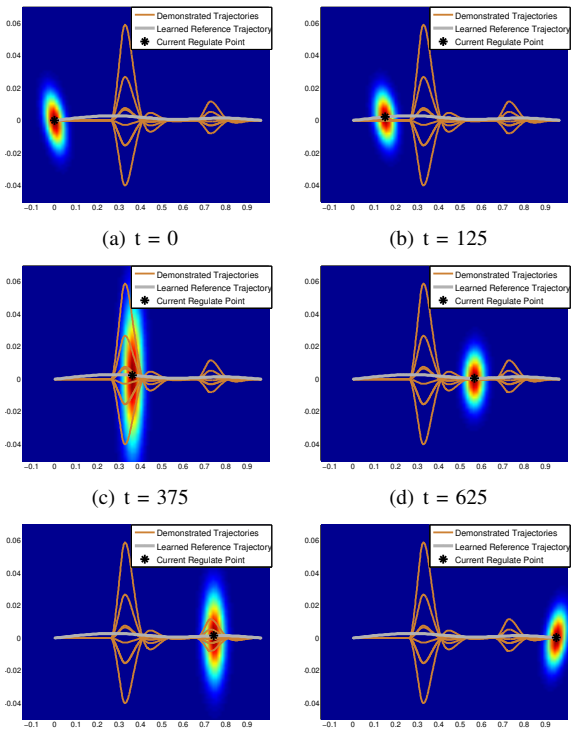
The dynamical system is described with parameters of mass, stiffness and damping as $M_p = 0.2$, $K_p = 200$, and $D_p = \sqrt{K_p}/2$, respectively. In total, 1000 time steps are simulated and perturbations are applied during $T_1 = [300, 350]$ and $T_2 = [700, 750]$ with noise $\mathcal{N}(0, 50)$ and $\mathcal{N}(0, 10)$, respectively. 8 trajectories are collected as demonstrations and the results are shown in Figure 6.

In Figure 6, the gray curve tracked by the regulation point is the resulting reference trajectory. It is not surprising that the trajectory is almost a straight line, as demonstrated trajectories are symmetrically perturbed. By evaluating the exponential values of minus cost over the whole 2D space, the weight matrix is visualized by the heating ellipse. As the cost is a quadratic form, the heating shape is actually an unnormalized Gaussian. The steep degree of slope indicates the sensitivity of the cost with respect to the deviance in corresponding directions. It is shown that the heating ellipse varies the length of the axis in the direction orthogonal to the reference trajectory. The axis is longer within the section where demonstrated trajectories are diverse. This indicates that varying $\boldsymbol{Q}_t$ tends to tolerate error such that perturbed trajectories are still of good quality in terms of the learned cost.

### B. Letter Trajectories

In this experiment, we apply Algorithm 1 to a more practical scenario that learns trajectories of handwritten letters. The purpose of this experiment is to extract from demonstrations an informative cost as the task representation. The cost will be further exploited to derive robot motion for reproducing the writing task.

The letter trajectories are collected from dataset [13]. Only position coordinates are considered, thus the data consists of a series of 2D coordinates. In the dataset, the trajectories are aligned to the same time horizon by curve fitting and subsampling. All letter coordinates are within a comparable range and defined with respect to the trajectory end points. See Figure 5 for typical demonstrated trajectories.

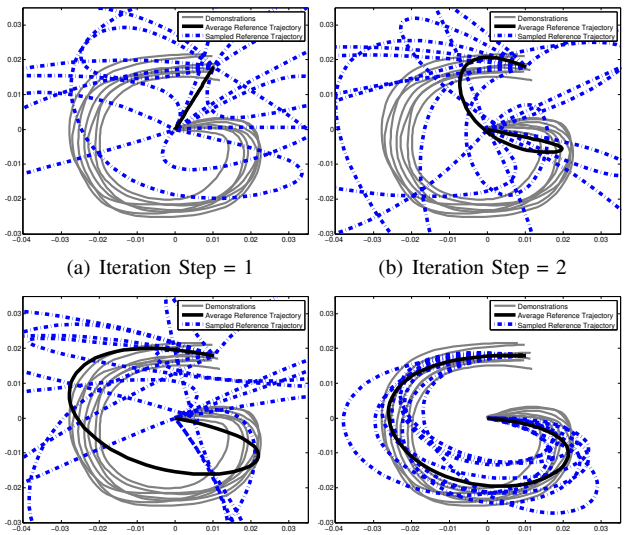



(a) t = 0 (b) t = 125

(c) t = 375 (d) t = 625

(e) t = 750 (f) t = 1000

Fig. 6. Result of simple particle example: reference trajectory and weight matrix as time evolves from 0 to 1000. The matrix $\boldsymbol{Q}_t$ is illustrated with heating ellipse by evaluating cost value over the whole state space.




(a) Iteration Step = 1 (b) Iteration Step = 2

(c) Iteration Step = 3 (d) Iteration Step = 10

Fig. 7. Evolution of reference trajectory as learning algorithm iterates from step 1 to 10. The iterations begin with a tentative straight line solution. The average trajectory evolves towards demonstrated profile to increase the likelihood of demonstrations. Sampled trajectories converge as covariance of sampling distribution shrinks at the final stage.

Figure 7 illustrates some particular iteration steps of the learning process for letter "G", where seven demonstrations are used as training data. The reference trajectory is naively initialized as a straight line, and the initial sampling distribution is set with a variance of 0.05 to ensure that a large enough parameter space is explored. The learned reference trajectory $\{\boldsymbol{x}_t^{ref}\}$ , which is encoded by the mean parameter of sampling distribution, rapidly converges to similar profile with demonstrated trajectories. The variance of the sampled trajectories decreases as the iteration evolves. This implies that the sampling distribution shrinks near to a Dirac function thus generated samples tend to be identical.

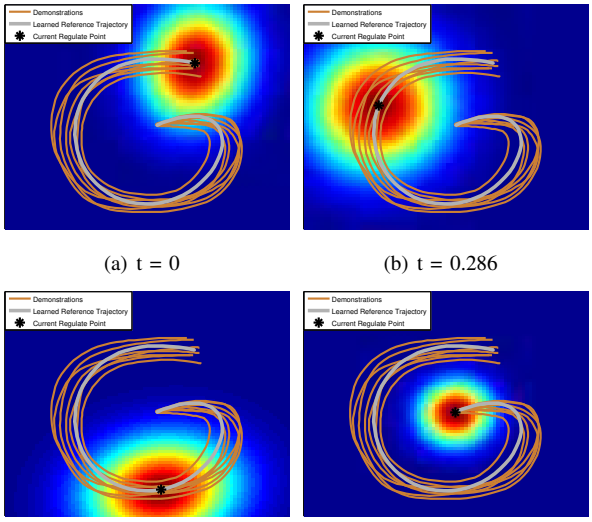
A more complete result for the letter "G" is shown in Figure 8. Here, the varying weight matrix $\boldsymbol{Q}_t$ is highlighted. The positive definite matrix is illustrated by a heating ellipse whose center is located at the current reference point, and the axes represent principle directions and weights. The direction of the principle axes varies as it is defined with respect to a local reference frame along the tracking trajectory. Also, it is observed that the length of principle axes, which indicates weight parameter in the corresponding direction, captures the sensitivity of deviance from reference trajectory at each regulation point. Similar to the simple synthetic example presented above, the ellipse expands its length of axis along the radial direction of the curve in 8(b), where demonstrated trajectories spread over a relatively larger space. On the contrary, in 8(c), the ellipse shrinks its axis length along the radial direction as the demonstrated trajectories are more consistent within these sections. Thus the deviance along this direction will incur a large cost penalty and the reference trajectory is expected to be well tracked. This indeed encodes a compliant behavior that the robot should adopt under disturbances. We will show that, based upon learned costs and optimal control, it is natural to develop varying stiffness behavior. The exploitation of motion feature $\boldsymbol{Q}_t$ will be further discussed in the following section.

### C. Decoding Motion on a Multi-fingered Hand

In this experiment, to derive writing motion on an anthropomorphic robot hand, we show an example of the use of Algorithm 2. We attempt to derive joint motion trajectories on a 16-DOFs Allegro multi-fingered hand (Figure 9). In the writing task, only three fingers (12 DOFs) are involved. One question about applying the learned cost is that we need to simulate a mapping function $\kappa$ to get features from the joint motion of multiple manipulators. We resolve this by employing a virtual frame that is commonly used in the grasping and dexterous manipulation community.

As shown in Figure 10, the virtual frame is statically defined by the position vector of the tips. For the case of three fingers, the origin ($O$ in Figure 10) of the virtual frame is the average position of involved end-effectors, and the orthogonal axes can be determined with the cross products of relative position vectors. The pen tip ($O'$ in Figure 10) is assumed to be fixed, with respect to this virtual frame via a known transformation. Note that $\kappa$ is designed for evaluating the cost and it is not known to the algorithm. We refer to [14] for more details about the definition of virtual frame and its application on multi-fingered manipulation task.

In practice, $N = 15$ samples are sufficient for exploring an optimal result. A cost that encodes writing motion of




(a) t = 0 (b) t = 0.286

(c) t = 0.571 (d) t = 1.0

Fig. 8. Result of learning letter "G": reference trajectory and weight matrix as time evolves. The matrix $\boldsymbol{Q}_t$ is illustrated with heating ellipse by evaluating cost value over the whole state space. The time horizon is scaled between 0 and 1.0.

Fig. 9. Allegro Multi-fingered Hand

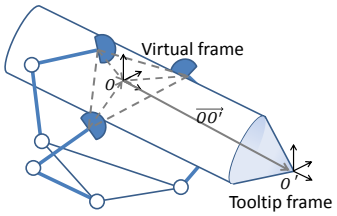



Fig. 10. Virtual frame and local tooltip reference point definition

letter "e" is used here. The candidate trajectories are again initialized as straight lines in the joint space. The evolution of cost values within 1000 iterations is shown in Figure 11. The cost monotonically decreases to a relatively stable level within a few hundred iterations. Also, the variability (gray area) of the costs of sampled trajectories decreases as the samples tend to be identical: implies that the exploration variance vanishes so that a convergence to a near optimal solution is achieved.

The decoded joint motion trajectory is applied on a simulated Allegro hand. The writing motion in simulation is shown as snapshots in Figure 12. To manipulate the orientation of the virtual frame and ensure letter profile is tracked, the generated motion exploits the redundancy of finger DOFs. The resultant trajectory is deposited from the movement of a tooltip that is rigidly attached to the virtual frame.

### *D. Exploiting $\boldsymbol{Q}_t$ in Developing Compliant Behavior*

Here we develop the compliant behavior for a robot by following the steps of the model-based optimal control. Concretely, we consider the end-effector motion in Cartesian

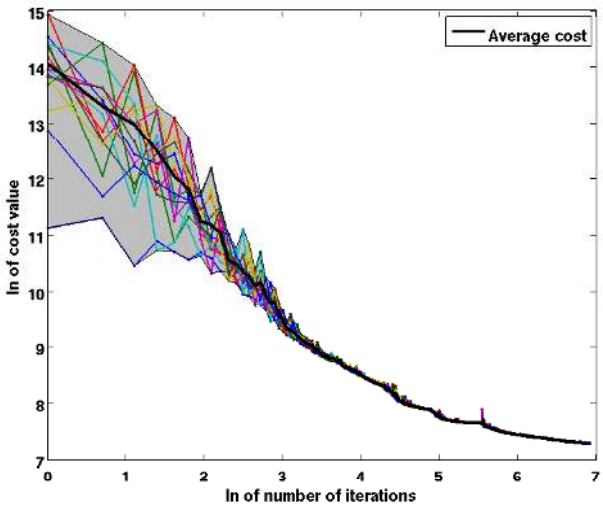



Fig. 11. Cost of sampled trajectories in throughout learning iterations

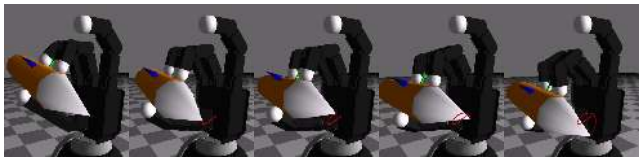
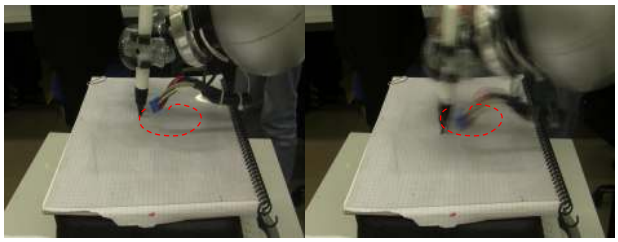


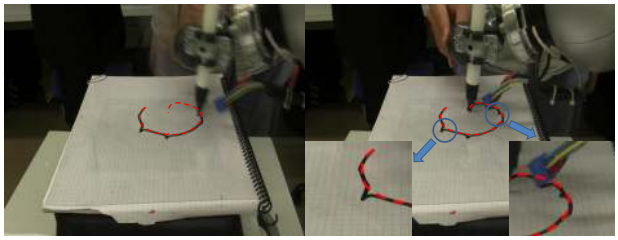
Fig. 12. Snapshots of generated writing motion on simulated Allegro hand

space of a 7-DOFs KUKA LWR robot. The derived trajectory and gain are realized with a Cartesian impedance controller. The encapsulated compliance is validated by subjecting the robot to disturbance during its writing execution. Figure 13 shows robot's compliant behavior with the developed varying impedance parameter. As the stiffness of robot is expected to co-vary with $\boldsymbol{Q}_t$, the robot exhibits relatively compliant behavior to perturbation, in Figure 13(b). We can compare this property with Figure 8(b). Note that in Figure 8, a smaller heating ellipse implies a larger $\boldsymbol{Q}_t$ as the evaluated values are shown as an unnormalized Gaussian. Similarly, the robot is comparatively stiff in the radial direction in Figure 13(c) and we observe even more resistance under perturbation in Figure 13(d). This is due to a larger $\boldsymbol{Q}_t$ in these sections thus increased impedance parameters are developed.

We also validate the generality of the learned weight matrices. As is shown in Figure 14, letters "N" and "W" are written with the impedance trajectory, which is derived by exploiting $\boldsymbol{Q}_t$ learned from "G". Because $\boldsymbol{Q}_t$ is locally defined along the reference trajectory, it encapsulates the knowledge of shaping stiffness ellipse to align with the direction of movement. This enables the robot to successfully track the modified trajectory by overcoming the friction, which is the main disturbance along the motion direction. It is expected to encapsulate more general task characteristics by incorporating other interesting features such as local geometrical parameters of the reference trajectory.



(a) Start writing (b) Large deviance under perturbation



(c) Small deviance under perturbation (d) End of writing

Fig. 13. Snapshots of motion of writing "G" with developed impedance parameters: (b) Low stiffness along radial direction - large deviance and vibration incurred under perturbation; (c) and (d) High stiffness - small oscillation amplitude under perturbation; Reference trajectory is illustrated as red dash line and the perturbed sections are shown in detail in (d). Note to compare with the shape of $\boldsymbol{Q}_t$ in Figure 8

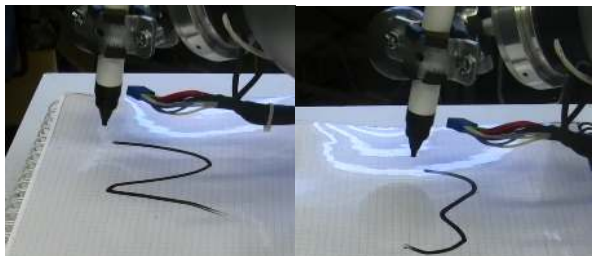


Fig. 14. Generalization of derived parameters to other letters: Writing "N" and "W" with impedance derived by exploiting $\boldsymbol{Q}_t$ learned from "G".

## V. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed two algorithms based upon the CE method for learning and reproducing robotic writing task. We have shown that the proposed cost encoding algorithm successfully learns the given parameterization, which is nonlinear with learning parameters. To decode the task cost by deriving a robot control policy, the algorithm for trajectory optimization has been validated as an effective approach on a challenging problem of planning motion for an anthropomorphic robotic hand. Moreover, the compliant behavior encapsulated in learned cost function is assessed in a robot writing experiment.

Note that the proposed algorithms are applicable to other forms of cost function, as the CE method imposes no restrictions on the exact form of objectivity to optimize. One aspect of the algorithm that needs improvement is the quality of the partition function evaluation in the cost learning. This is a kind of regularization and a common computational challenge to the general probabilistic inference problem. We expect to increase the performance of the proposed algorithm by introducing advanced methods for efficient calculation of the partition function.

Another possible extension would be employing sampling distributions other than a single Gaussian. In [7] and [15], mixture of Gaussians is used to guide the policy search. Exploring in a richer family of distributions promises more accurate rare-event probability estimation. This is expected as a better sampling schema, though it also leads to a non-trivial parameter update in the KL divergence minimization.

Finally, the proposed methods are open to the incorporation of intrinsic and environment dynamics, that can encapsulate more task-specific knowledge for the transfer of underlying skills.

## ACKNOWLEDGMENT

This work is partially funded by Swiss National Center of Robotics Research and an FCT doctoral grant (SFRH/BD/51933/2012) under IST-EPFL Joint Doctoral Initiative.

## REFERENCES

[1] S.M. Khansari-Zadeh, Klas Kronander, and Aude Billard. Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 26(17):1967–1993, 2012.

[2] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, pages 323–329, Osaka, Japan, 2012.

[3] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *In IEEE International Conference on Robotics and Automation (ICRA2002*, pages 1398–1403, 2002.

[4] Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. Maximum margin planning. In *International Conference on Machine Learning*, July 2006.

[5] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[6] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

[7] M. Kobilarov. Cross-entropy motion planning. *International Journal of Robotics Research*, 31(7):855–871, 2012.

[8] Freek Stulp and Pierre-Yves Oudeyer. Adaptive exploration through covariance matrix adaptation enables developmental motor learning. *Paladyn*, 3(3):128–135, 2012.

[9] Elmar A. Rückert, Gerhard Neumann, Marc Toussaint, and Wolfgang Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience*, 6(97), 2013.

[10] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1049–1056, New York, NY, USA, 2009. ACM.

[11] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[12] Hilbert J. Kappen, Vicen Gmez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

[13] S. M. Khansari-Zadeh and Aude Billard. Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. *IEEE Transaction on Robotics*, 2011.

[14] M. Li, H. Yin, K. Tahara, and A. Billard. Learning object-level impedance control for robust grasping and dexterous manipulation. In *Proceedings of International Conference on Robotics and Automation (ICRA), 2014*, Hong Kong, China.

[15] S. Calinon, A. Pervez, and D. G. Caldwell. Multi-optima exploration with adaptive gaussian mixture model. In *Proc. Intl Conf. on Development and Learning (ICDL-EpiRob)*, San Diego, USA, 2012.