Learning Deep Robot Controllers by Exploiting Successful and Failed Executions

Domingo Esteban[1,2], Leonel Rozo[1] and Darwin G. Caldwell[1]

***Abstract*— The prohibitively amount of data required when learning complex nonlinear policies, such as deep neural networks, has been significantly reduced with guided policy search (GPS) algorithms. However, while learning the control policy, the robot might fail and therefore generate unacceptable guiding samples. Failures may arise, for example, as a consequence of modeling or environmental uncertainties, and thus unsuccessful interactions should be explicitly considered while learning a complex policy. Currently, GPS methods update the robot policy discarding or giving low probability to unsuccessful trials. In other words, these methods overlook the existence of poorly performing executions, and therefore tend to underestimate the information of these interactions in next iterations. In this paper we propose to learn deep neural network controllers with an extension of GPS that considers trajectories optimized with *dualist constraints*. These constraints are aimed at assisting the policy learning so that the trajectory distributions updated at each iteration are similar to *good* trajectory distributions (e.g., sucessful executions) while differing from *bad* trajectory distributions (e.g. failures). We show that neural network policies guided by trajectories optimized with our method reduce the failures during the policy exploration phase, and therefore encourage safer interactions. This may have a relevant impact in tasks that involve physical contact with the environment or human partners.**

I. INTRODUCTION

The complexity and variety of tasks that a humanoid robot may face in real scenarios, where environments are usually dynamic and unstructured, require a high degree of autonomy. For this reason, the idea to autonomously discover optimal behaviors from experience has motivated the research community to make reinforcement learning (RL) feasible in high-dimensional, continuous or partially-observed spaces. In this context, the application of RL in robotics has been aimed at reducing the complexity of these problems by exploiting policy search (PS) methods with domain-appropriate pre-structured policies, which has provided faster learning in simple scenarios [1]. Specifying the policy through general-purpose representations such as neural networks (NN) has demonstrated to successfully model complex behaviors and to work directly with raw sensory data in end-to-end frameworks [2][3]. Nevertheless, the application of these complex policies in robotics is still limited because the majority of PS methods either do not scale well with high-dimensional policies or require a prohibitively amount of interaction with the environment. On the other hand, guided policy search (GPS) algorithms are data-efficient methods for learning NN policies because they transform the policy search problem into supervised learning, where the training data is generated by a computational teacher that produces data that is best suited for training the final policy. Simple trajectory-centric RL algorithms [4] or complex trajectory optimization methods [5] showed the benefit of such guiding policies.

As with any other RL agent, the experiences that a robot obtains while learning a task are not always good or successful. For example, an aggressive exploration, a flawed definition of the robot/environment state, the existence of limited useful data or the stochasticity of the environment may lead the robot to fail and therefore generate bad or undesirable rollouts. Nonetheless, most of the policy update strategies in RL do not completely take advantage of this information and are limited to increase the probability of occurrence of high reward samples and give only low probability to the failures, which makes the robot forget the unsuccessful executions in subsequent iterations [1][6].

This problem can be even more critical in GPS, because the guiding policies seek to minimize an expected surrogate cost that includes not only the task-related cost, but also a term that encourages the guiding policies to resemble the complex global policy. The latter term may lead the guiding policies to produce undesired events that generate a sudden increment in the cost values, which we necessarily want to avoid in the following iterations. Our desire to avoid these failures can, to a certain extent, be captured by a well-designed cost function. However, as in the case of mirror descent guided policy search (MDGPS) [7], the change of the guiding policies is limited by a Kullback-Leibler (KL) divergence constraint, and therefore samples generated by these updated policies may produce the same undesired failures of previous iterations. Additionally, the supervision step in GPS uses all the trajectory samples that were generated by the guiding policies, therefore fitting the global policy to trajectories that, in fact, we are trying to avoid. Consequently, next guiding policies will be constrained to be similar to a global policy that was trained with flawed samples.

In this paper we propose a model-based trajectory-centric RL algorithm that explicitly exploits good and bad experiences in the policy optimization step. First, good and bad trajectory distributions are defined and updated by samples with low and high cost, respectively. Then, similarly to [8], these trajectories are explicitly considered in the policy update by including upper bounds on the KL divergences between the distributions of new and good trajectories, and lower bounds on the KL divergences between new and bad trajectory distributions. The trajectory distributions optimized with these

[1]Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy name.surname@iit.it

[2]DIBRIS, Università di Genova, Via Opera Pia 13, 16145, Italy

dualist constraints can be used in GPS to guide the learning process of a complex policy representation, therefore leading to global policies that encourage safer interactions.

Related work and background are given in Sec. II and III, respectively. The details of the proposed RL algorithm are presented in Section IV-A and the extension of MDGPS that considers samples generated by our dualist-optimized trajectories is introduced in Section IV-B. The experiments in Section V are aimed at demonstrating how a policy guided by our algorithm decreases the number of unsuccessful executions compared to policies guided by previous model-based RL methods. A discussion of the reported results and future research work is given in Section VI.

II. RELATED WORK

The possibility to use robust algorithms and simple local policies with few parameters has made GPS a popular framework to learn complex policies. The simple local policies are employed as computational teachers that generate guiding distributions for a nonlinear global policy. Such simple policies are usually trajectory-centric representations such as splines, dynamic movement primitives [9] and time-varying linear-Gaussian (TVLG) controllers. The latter are popular in stochastic optimal control and various trajectory optimization methods, such as the iterative linear quadratic Gaussian (iLQG) algorithm [10][11]. In this paper we also exploit the properties of TVLG controllers to represent the simple policies that generate the trajectory distributions from both successful and failed executions. Typically the algorithms used to optimize TVLG controllers assume a known (or iteratively learned) dynamics model [12][13].

None of the aforementioned algorithms exploit the existence of poorly performing samples. In PI-GPS [14], the samples with high cost-to-go values are practically ignored if better samples are obtained in the same iteration because of the assigned low-probability scores. Then, the high-cost samples are barely considered in subsequent policy updates. Note that giving this treatment to failed executions may generate policy updates that still lead to failures in the next few iterations. Nevertheless, learning from failures is a promising approach to discover successful and safer ways to accomplish tasks. If we explicitly consider a dataset of negative/undesired samples, we can exclude regions in the parameter space that lead to failures by favoring regions that conduce to successful executions [15][16].

Notice that in inverse reinforcement learning (IRL), considering negative state-action trajectories generates more accurate reward functions than methods relying on positive trajectories exclusively. In [17], the authors included negative demonstrations into the optimization of a maximum-causal-entropy IRL method. In such a way, the method obtained, in less iterations, linear rewards functions that generalized better even when the successful and failed demonstrations were contrasting, overlapping, or complementary. Choi *et al.* [18] proposed to use a Gaussian process to represent a nonlinear reward function whose kernel moved the prediction close to positive samples and drift it away from negative ones. Beyond the fact that our work addresses a policy search problem, the main difference of the aforementioned approaches with respect to this paper is that the agent does not have access to previous datasets of positive and negative trajectories, but these are instead obtained during its interaction with the environment and latter classified as good or bad samples based on their accumulated cost.

Our approach is inspired by the dual relative entropy policy search (DREPS) [8], a generalization of REPS [19] that takes into account both good and bad samples when computing a policy. In DREPS, a closed form update of the parameters is obtained from the Lagrangian of an optimization problem that bounds the KL divergence between the new policy and precomputed low- and high-performance clusters of parameters. Unlike DREPS, our algorithm is formulated in a step-based policy search setting, which means that our method has notion of state-space and sequential decisions. In addition to this, our work is built on model-based RL, where a (local) dynamics model is learned before carrying out the policy optimization step. Lastly, our method does not directly update the robot policy parameters, but it instead exploits MDGPS to first optimize simple low-dimensional policies using the proposed dualist constraints, and subsequently uses the resulting guiding trajectories to update the parameters of a complex global policy through supervised learning.

III. PRELIMINARIES AND OVERVIEW

We consider an episodic learning problem for a time horizon T, where the robot behavior is defined by a parametrized stochastic policy $\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$, which is a distribution over actions $\mathbf{u}_t$ conditioned on the state $\mathbf{x}_t$. The goal of PS is to optimize the parameters $\boldsymbol{\theta}$ with respect to the expected cost $\mathbb{E}_{p(\tau)}\left[\ell(\tau)\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\sum_{t=1}^{T}\ell(\mathbf{x}_t,\mathbf{u}_t)]$. The expectation is computed with respect to the trajectory $\tau = (\mathbf{x}_1,\mathbf{u}_1,\ldots,\mathbf{x}_T,\mathbf{u}_T)$, whose distribution is induced by the policy $\pi_{\boldsymbol{\theta}}$ and the system dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)$:

$$\pi_{\boldsymbol{\theta}}(\tau) = p(\mathbf{x}_1)\prod_{t=1}^{T}\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t),$$

where $p(\mathbf{x}_1)$ is the initial state distribution.

A. Model-based trajectory-centric reinforcement learning

The iterative linear quadratic Gaussian algorithm (iLQG) [10][11] is an efficient indirect method for trajectory optimization. Unlike differential dynamic programming (DDP), iLQG only uses the first derivative of the discrete-time dynamics, thus allowing a faster dynamics evaluation that outweighs the decrease in performance. The algorithm considers a quadratic expansion of the cost around a nominal trajectory $\hat{\tau} = (\hat{\mathbf{x}}_1,\hat{\mathbf{u}}_1,\ldots,\hat{\mathbf{x}}_T,\hat{\mathbf{u}}_T)$, and a local linear-Gaussian approximation of the dynamics $\mathcal{N}(f_{\mathbf{x}t}\mathbf{x}_t + f_{\mathbf{u}t}\mathbf{u}_t + f_{ct},\mathbf{F}_t)$ with a covariance $\mathbf{F}_t$, and a mean given by the gradients $f_{\mathbf{x}t}$ and $f_{\mathbf{u}t}$, and a constant term f_{ct}. Under these assumptions, the first and second derivatives of the state-value and action-value functions, namely $V(\mathbf{x}_t)$ and $Q(\mathbf{x}_t,\mathbf{u}_t)$ respectively,

are:

$$
\begin{aligned}
Q_{\mathbf{xu},\mathbf{xu}t} &= \ell_{\mathbf{xu},\mathbf{xu}t} + f_{\mathbf{xu}t}^{\mathsf{T}} V_{\mathbf{x},\mathbf{x}t+1} f_{\mathbf{xu}t} \\
Q_{\mathbf{xu}t} &= \ell_{\mathbf{xu}t} + f_{\mathbf{xu}t}^{\mathsf{T}} V_{\mathbf{x}t+1} \\
V_{\mathbf{x},\mathbf{x}t} &= Q_{\mathbf{x},\mathbf{x}t} - Q_{\mathbf{u},\mathbf{x}t}^{\mathsf{T}} Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u},\mathbf{x}t} \\
V_{\mathbf{x}t} &= Q_{\mathbf{x}t} - Q_{\mathbf{u},\mathbf{x}t}^{\mathsf{T}} Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u}t}
\end{aligned}
$$

where the derivatives are denoted by subscripts, so for example $\ell_{\mathbf{xu}t}$ is the gradient of the cost at time step t with respect to $[\mathbf{x}, \mathbf{u}]^{\mathsf{T}}$ and $\ell_{\mathbf{xu},\mathbf{xu}t}$ the Hessian. The optimal linear control law that minimizes $Q(\mathbf{x}_t, \mathbf{u}_t)$ is $g(\mathbf{x}_t) = \hat{\mathbf{u}}_t + \mathbf{k}_t + \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t)$, where $\mathbf{K}_t = -Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u},\mathbf{x}t}$ and $\mathbf{k}_t = -Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u}t}$.

A Gaussian trajectory distribution $p(\tau)$ can be obtained by considering a linear-Gaussian controller $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{C}_t)$. The mean of this distribution is given by the above deterministic solution assuming, for notational convenience, that the nominal states and actions are zero. The covariance $\mathbf{C}_t$ is proportional to the curvature of the Q-function, $\mathbf{C}_t = Q_{\mathbf{u},\mathbf{u}t}^{-1}$. Note that this linear-Gaussian controller also optimizes the maximum entropy objective as shown in [12], which is formulated as:

$$
\begin{aligned}
p(\tau) \leftarrow \underset{p(\tau)\in\mathcal{N}(\tau)}{\arg\min}\ & \mathbb{E}_{p(\tau)}\left[\ell(\tau)\right] - \mathcal{H}(p(\tau)) \\
s.t.\ & p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f_{\mathbf{x}t}\mathbf{x}_t + f_{\mathbf{u}t}\mathbf{u}_t + f_{ct}, \mathbf{F}_t)
\end{aligned}
$$

When the dynamics is unknown, a distribution $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ can be estimated around the trajectories sampled from the real system under the previous linear-Gaussian controller $\hat{p}(\mathbf{u}_t|\mathbf{x}_t)$. To prevent the dynamic programming pass in the iLQR from drastically modifying the new controller, which makes the local dynamics invalid around the new trajectory distribution, the policy update should be constrained. Levine & Abbeel [12] proposed to include a KL divergence constraint on the previous trajectory distribution $\hat{p}(\tau)$, as follows:

$$
\min_{p(\tau)} \mathbb{E}_{p(\tau)}[\ell(\tau)] \quad \text{s.t.} \quad D_{\mathrm{KL}}(p(\tau)||\hat{p}(\tau)) \leq \epsilon, \tag{1}
$$

where ϵ denotes the maximal information loss, and the dynamics constraint has been omitted for clarity. The Lagrangian of (1) is

$$
\mathcal{L}(p(\tau), \eta) = \mathbb{E}_{p(\tau)}[\ell(\tau)] + \eta[D_{\mathrm{KL}}(p(\tau)||\hat{p}(\tau)) - \epsilon],
$$

and since $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \hat{p}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, the resulting Lagrangian becomes

$$
\begin{aligned}
\mathcal{L}(p(\tau), \eta) = &\left[\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \eta \log \hat{p}(\mathbf{x}_t, \mathbf{u}_t)]\right] \\
&- \eta\mathcal{H}(p(\tau)) - \eta\epsilon
\end{aligned} \tag{2}
$$

The above problem is solved in [12] by dual gradient descent, alternating between a dynamic programming pass to minimize the Lagrangian with respect to $p(\tau)$, and adjusting η according to the amount of constraint violation.

B. Guided policy search algorithms

Complex nonlinear models, such as neural networks, allow policies to model complex behaviors and permit to work directly with sensory input data [2][3][4], at the expense of learning thousands of parameters. Despite the important advances in model-based and model-free PS methods [1], their application is generally limited to specific policy representations with less than a hundred parameters, or require a prohibitively amount of interactions with the environment [2]. Instead of optimizing the parameters directly from the expected cost, GPS methods transform the policy search problem into supervised learning, where the training set is generated by a computational teacher, optimized by either simple trajectory-centric RL algorithms [4] or complex trajectory optimization methods [5]. In such a way, the convergence of a global policy that minimizes the expected cost is obtained by solving

$$
\min_{\boldsymbol{\theta}, p(\tau)} \mathbb{E}_{p(\tau)}[\ell(\tau)] \quad \text{s.t.} \quad p_i(\mathbf{u}_t|\mathbf{x}_t) = \pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t) \quad \forall t, \forall i, \tag{3}
$$

meaning that the learning process of the global policy is indeed divided into a domain-specific optimization of local policies $p_i(\mathbf{u}_t|\mathbf{x}_t)$, and a supervised phase for the global policy $\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$ so that it matches the simple policies. When $p_i(\mathbf{u}_t|\mathbf{x}_t)$ is represented by a TVLG controller, the method described in Section III-A can be used.

The constrained minimization problem in (3) guarantees that the global policy minimizes the expected cost at convergence, but barely focus on the robot behavior in intermediate iterations. This limitation is solved by MDGPS [7], which approximates a local TVLG controller $\bar{\pi}_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$ to the global policy $\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$, by reformulating the local policy constraint in (1) as (under linearity and convexity assumptions):

$$
\min_{p(\tau)} \mathbb{E}_{p(\tau)}[\ell(\tau)] \quad \text{s.t.} \quad D_{\mathrm{KL}}(p(\tau)||\bar{\pi}_{\boldsymbol{\theta}}(\tau)) \leq \epsilon, \tag{4}
$$

where $\bar{\pi}_{\boldsymbol{\theta}}(\tau)$ is the trajectory induced by $\bar{\pi}_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$.

Note that the global policy learned by GPS exclusively focus on behaviors provided by the guiding distributions $p_i(\tau)$, meaning that it ignores completely how these trajectories were obtained. Thus, if we require the complex global policy to consider bad behaviors during the learning process, the local policies updates must explicitly consider the undesired high-cost experiences into the optimization problem. In such a way, the policy search is expected to avoid policy parameters that generate unsafe or unsuccessful executions, therefore being less prone to failures.

IV. DEEP REINFORCEMENT LEARNING WITH DUALIST UPDATES

First of all, let us define $\mathcal{G}$ and $\mathcal{B}$ as the sets representing good and bad experiences, respectively. In a policy search setting, these experiences are encoded by good and bad trajectory probability distributions that are generated from successful (low cost) or failed (high cost) task executions. These trajectory distributions can be explicitly considered in

the policy update by including an upper bound on the KL divergence between the new and good trajectory distributions, and a lower bound on the KL divergence between the new and bad trajectory distributions. In this way, similarly to [8], we reformulate the policy update as follows

$$\begin{aligned} \boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\tau)}\left[\ell(\tau)\right] \\ s.t.\ D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(\tau)||g_c(\tau)) \leq \chi,\ c \in \mathcal{G} \\ D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(\tau)||b_d(\tau)) \geq \xi,\ d \in \mathcal{B} \end{aligned} \tag{5}$$

where $g_c(\tau)$ are the good trajectory distributions in $\mathcal{G}$, $b_d(\tau)$ are the bad trajectory distributions in $\mathcal{B}$, χ the maximal information loss with respect to the good trajectory distributions and ξ the minimal information loss with respect to the bad trajectory distributions. The interpretation of these dualist constraints is very intuitive: our policy learns parameters that generate trajectories that differ from trajectory distributions that induce bad behaviors while resembling trajectories that lead to successful executions.

Note that when the policy is represented by a deep neural network, the optimization process using typical RL methods is infeasible due to the high dimensionality of the policy parameters. Therefore, we propose to use GPS (described in Section III-B) and generate guiding local policies to assist and accelerate the policy search process. In this sense, GPS needs to be reformulated in order to include dualist constraints as explained next.

A. Model-based (trajectory-centric) RL with dualist updates

Without loss of generality, both good and bad behaviors are defined by a single trajectory distribution each. Then, the cardinalities of the good and bad sets are $|\mathcal{G}| = |\mathcal{B}| = 1$. Let us define good and bad policies as $g(\mathbf{u}_t|\mathbf{x}_t)$ and $b(\mathbf{u}_t|\mathbf{x}_t)$, that induce a good trajectory distribution $g(\tau)$ and a bad trajectory distribution $b(\tau)$, respectively. Therefore, the policy update $p(\tau)$ can be reformulated as

$$\begin{aligned} p(\tau) \leftarrow \arg\min_{p(\tau)} \mathbb{E}_{p(\tau)}[\ell(\tau)] \\ s.t.\ D_{\mathrm{KL}}(p(\tau)||\hat{p}(\tau)) \leq \epsilon \\ D_{\mathrm{KL}}(p(\tau)||g(\tau)) \leq \chi \\ D_{\mathrm{KL}}(p(\tau)||b(\tau)) \geq \xi. \end{aligned} \tag{6}$$

The Lagrangian of (6) is defined as

$$\begin{aligned} \mathcal{L}(p(\tau),\eta,\omega,\nu) =& \mathbb{E}_{p(\tau)}[\ell(\tau)] + \eta[D_{\mathrm{KL}}(p(\tau)||\hat{p}(\tau)) - \epsilon] \\ &+ \omega[D_{\mathrm{KL}}(p(\tau)||g(\tau)) - \chi] \\ &+ \nu[\xi - D_{\mathrm{KL}}(p(\tau)||b(\tau))], \end{aligned} \tag{7}$$

where η, ω and ν are the Lagrange multipliers controlling the relevance of each inequality constraint in (6).

Due to the linear-Gaussian dynamics assumption, the minimization of (7) with respect to $p(\tau)$ can be written as:

$$\begin{aligned} \min_{p(\tau)} \quad & \mathbb{E}_{p(\tau)}\left[\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x}_t,\mathbf{u}_t)}[\tilde{\ell}(\mathbf{x}_t,\mathbf{u}_t)]\right] - \mathcal{H}(p(\tau)) \\ & - \frac{1}{\eta+\omega-\nu}\left[\eta\epsilon + \omega\chi - \nu\xi\right] \end{aligned} \tag{8}$$

Algorithm 1 Dualist GPS

1: Initialize p_i
2: **for** iteration $k = 1$ to K **do**
3: Run p_i to collect trajectory samples $\mathcal{D}_i = \{\tau_i\}$
4: Fit linear-Gaussian dynamics $p_i(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ to $\mathcal{D}_i$
5: Fit linearized global policy $\bar{\pi}_{\boldsymbol{\theta}_i}(\mathbf{u}_t|\mathbf{x}_t)$ to $\mathcal{D}_i$
6: Update $g(\tau)$ and $b(\tau)$ from $\mathcal{D}_i$
7: Adjust ϵ
8: Optimize p_i from Equation (9)
9: Optimize $\pi_{\boldsymbol{\theta}}$ from Equation (13)
10: **end for**

where

$$\tilde{\ell}(\mathbf{x}_t,\mathbf{u}_t) = \tfrac{1}{\eta+\omega-\nu}\left[\ell(\mathbf{x}_t,\mathbf{u}_t) - \eta\log\hat{p}(\mathbf{u}_t|\mathbf{x}_t) - \omega\log g(\mathbf{u}_t|\mathbf{x}_t) + \nu\log b(\mathbf{u}_t|\mathbf{x}_t)\right].$$

Then, considering this augmented cost $\tilde{\ell}(\mathbf{x}_t,\mathbf{u}_t)$, the primal problem in (6) can also be solved using (8) with a process similar to that described in Section III-A.

B. Dualist GPS

Learning a complex nonlinear policy by solving problem (5) usually requires a large amount of interactions, where failed (possible dangerous) executions might arise until an acceptable suboptimal policy is obtained. These failures may lead to serious consequences on the robot and the environment it interacts with. Therefore, by reformulating the GPS framework to include dualist constraints, we can reduce not only the number of data required to learn $\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$, but also the robot failures during the learning process.

As mentioned in Section III-B, a global policy learned by the classical GPS optimizes its parameters based only on the behaviors provided by the guiding distributions $p_i(\tau)$. Thus, using trajectories optimized with our trajectory-centric RL algorithm, implies that the global policy $\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_t)$ considers both good and bad experiences, as outlined in Algorithm 1.

Our algorithm employs MDGPS to learn the global policy. Therefore, the optimization problem (6) for each local policy i is rewritten as

$$p_i(\tau) \leftarrow \arg\min_{p_i} \mathbb{E}_{p_i(\tau)}\left[\sum_{t=1}^{T} \ell(\mathbf{x}_t,\mathbf{u}_t)\right] \tag{9}$$

$$s.t.\ D_{\mathrm{KL}}(p_i(\tau)||\bar{\pi}_{\boldsymbol{\theta}_i}(\tau)) \leq \epsilon \tag{10}$$

$$D_{\mathrm{KL}}(p_i(\tau)||g_i(\tau)) \leq \chi \tag{11}$$

$$D_{\mathrm{KL}}(p_i(\tau)||b_i(\tau)) \geq \xi \tag{12}$$

where, $\bar{\pi}_{\boldsymbol{\theta}_i}(\tau)$ is the trajectory induced by the local TVLG approximation $\bar{\pi}_{\boldsymbol{\theta}_i}(\mathbf{u}_t|\mathbf{x}_t)$. Note that good $g_i(\tau)$ and bad $b_i(\tau)$ distributions are defined for each $p_i(\tau)$, and updated by samples generated by their respective TVLG controller.

The supervision step in MDGPS is the same as the original formulation, meaning that the global policy is obtained by:

$$\pi_{\boldsymbol{\theta}} \leftarrow \arg\min_{\boldsymbol{\theta}} \sum_{t,i,j} D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(\mathbf{u}_t|\mathbf{x}_{t,i,j})||p_i(\mathbf{u}_t|\mathbf{x}_{t,i,j})) \tag{13}$$

C. Defining good and bad experiences

Classifying a sampled trajectory either as good or bad, clearly involves an assessment based on different, possibly many, criteria. If the learning process is carried out under the supervision of a human, then the human may provide feedback regarding the goodness of every robot execution. But in autonomous settings, our expectations of what we want the robot *to do* and *not to do* are mainly captured by the cost function. The main goal of a robot in an RL problem is to minimize the expected cost. For this reason, bad experiences are here defined as undesirable trajectories that have a high expected cost. Similarly, good experiences are trajectories with low expected cost. In such a way, at each iteration k, n_g good samples and n_b bad samples are used to update the trajectories in $\mathcal{G}$ and $\mathcal{B}$, respectively. Note that these definitions rely on the fact that the cost function captures not only how the robot behavior should be to perform optimally, but also how it should never be.

As we mentioned previously, we assume that the cardinalities of the good and bad sets are $|\mathcal{G}| = |\mathcal{B}| = 1$. This involves that all the n_g samples with lower cost are used to update a single good trajectory distribution $g(\tau)$ and all the n_b samples with high cost are used to update a single bad trajectory distribution $b(\tau)$. However, there are different ways to construct the distributions $g(\tau)$ and $b(\tau)$, in this paper we obtain them by following the same method employed in MDGPS to get the local TVLG approximation $\bar{\pi}_{\boldsymbol{\theta}}(\tau)$. Specifically, at each iteration k, we fit a TVLG controller to the observed good trajectories and another TVLG controller to the observed bad trajectories.

V. EXPERIMENTS

The proposed framework was evaluated in two reaching tasks with collision avoidance. Both a simulated 3-DoF planar manipulator and a simulated humanoid robot were required to reach a desired Cartesian pose while avoiding an obstacle that is halfway.

A. Planar manipulator robot

1) Description: In the first experiment, the RL agent is a 3-DoF planar manipulator as shown in Fig. 1. The state of the task is defined by $\mathbf{x} \in \mathbb{R}^{12}$, composed of joint positions $\mathbf{q} \in \mathbb{R}^3$, joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^3$, and end-effector pose relative to the target $\mathbf{g} \in \mathbb{R}^3$ and to the obstacle $\mathbf{o} \in \mathbb{R}^3$. The action $\mathbf{u} \in \mathbb{R}^3$ corresponds to the robot torque commands.

The cost function that evaluates the performance of the task execution was defined as:

$$\ell(\tau) = \ell(\mathbf{x}_T, \mathbf{u}_T) + \sum_{t=1}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) \tag{14}$$

with final cost

$$\ell(\mathbf{x}_T, \mathbf{u}_T) = w_1(\gamma_1 + ||\mathbf{g}_T||^2)^{1/2} + w_2||\mathbf{g}_T||^2 + w_3 \text{max}(d_{\text{SAFE}} - d(\mathbf{o}_T), 0),$$

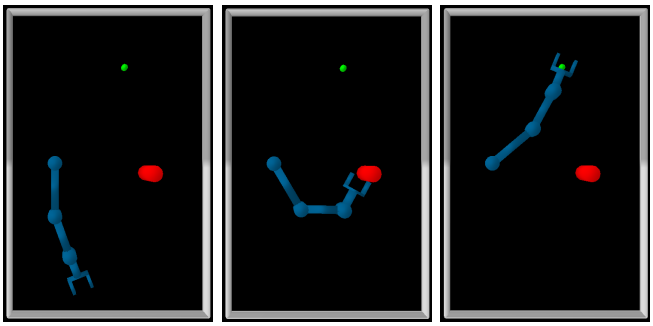

Fig. 1: Reaching task of a planar manipulator. The robot should learn to reach a target Cartesian pose (depicted in green) without touching an obstacle (red cylinder). During the iterative learning process, the robot may collide with the obstacle generating high-cost executions that are considered failures. Such failed executions are exploited by our dualist GPS to provide safer global policies.

and stage cost

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = w_4(\gamma_2 + ||\mathbf{g}_t||^2)^{1/2} + w_5||\mathbf{g}_t||^2 + w_6 \text{max}(d_{\text{SAFE}} - d(\mathbf{o}_t), 0) + w_7||\mathbf{u}_t||^2$$

where $w_1 = 1.0 \times 10^4$, $w_2 = 50.0$, $w_3 = 2.0 \times 10^6$, $w_4 = 10.0$, $w_5 = 5.0 \times 10^{-2}$, $w_6 = 2.0 \times 10^3$, $w_7 = 1.0 \times 10^{-6}$, $\gamma_1 = 10^{-10}$, $\gamma_2 = 10^{-10}$, $d_{\text{SAFE}} = 0.15$, and $d(\mathbf{o})$ is the signed distance evaluated on $\mathbf{o}$. The cost term $\max(d_{\text{SAFE}} - d(\mathbf{o}_t), 0)$ is a hinge loss that does not penalize the robot if the distance $\mathbf{o}$ is further than d_{SAFE} and favors collision avoidance behaviors [20].

The global policy consisted of a fully connected feed-forward neural network with two hidden layers of 40 rectified linear units each. Four Cartesian target and obstacle poses, defined by the 2D position and orientation with respect the vertical axis, were randomly generated. Therefore, four TVLG controllers ($i = 4$) were iteratively optimized as guiding policies.

The experiments carried out in this scenario were focused on analyzing how many times and how fast the robot approached, or even collided, with the obstacle, during the iterative learning process. First, we carried out an experiment aimed at observing the effect of discarding the worst samples in the supervised learning (SL) step of MDGPS. As a result the NN is trained to match a smaller dataset but only composed of non-bad trajectories. This intuitive strategy may be justified by the fact that we require the global policy to not reproduce the trajectories that generated high-cost. This strategy is in some way similar to that followed by most of PS methods, which discard or give low probability to unsuccessful trials.

Latter, we compared three different GPS formulations. The first one did not consider neither (11) nor (12) constraints, hence corresponding to the standard MDGPS. The second formulation only took into account bad experiences, which means that (11) was not considered in the policy updates. The third formulation corresponded to the full dualist approach

proposed in this paper, which exploits both good and bad experiences. These formulations are here referred to as MDGPS, B-MDGPS and D-MDGPS, respectively. All of them were run with 50 iterations using the same set of hyper-parameters, except by those related with the dualist constraints. Six trajectories were sampled at each iteration from every local policy, and subsequently the sample with lowest cost was used to update the good trajectory distribution, while the sample with highest cost was used to update the bad trajectory distribution ($n_g = 1$ and $n_b = 1$). Both dualist trajectory distributions had a fixed covariance. In order to carry out fair comparisons, the noise required by the whole exploration phase was identical for all the three aforementioned cases.

2) Results: Figure 2 shows the effect of discarding the worst samples in the SL step of MDGPS. The bottom plot displays the accumulated safe-distance cost, $\sum_{t=1}^{T} \max(d_{\text{SAFE}} - d(\mathbf{o}_t), 0)$, for the training conditions, where high values indicate that the resulting trajectory generated a dangerous robot end-effector movement as it was very close to the obstacle. Thus, the higher the value, the worst the trajectory, and therefore the more important it is to avoid trajectory distributions that generate this undesired behavior. Note that despite we identified the negative samples that generated the higher values of this cost and discarded them as training data for the SL step, the neural network policy (and subsequently the trajectory distributions that guided it) continued passing near the obstacle. Only after some more iterations the updated trajectories resulted in end-effector movements that avoided the obstacle.

On the other hand, the top plot of Fig. 2 shows that discarding the bad samples causes the robot to require more iteration to reach the desired pose, therefore, it has a negative impact on the main objective of the task. This effect may arise as consequence of the reduction of the training dataset used to train the neural network, which deteriorated its performance. Again, only after some more iterations (and no more bad trajectories), the final performance was the same as the standard MDGPS.

As stated previously, our main motivation to learn from bad experiences is to reduce failures during training. Figure 3 shows the accumulated safe-distance cost generated during the exploration phase. Observe that the formulation with only bad experiences and our approach iterated more safely during the exploration when compared to the classic MDGPS. Note that B-MDGPS tended to produce fewer failures, in other words, it generated fewer undesired end-effector movements that led to collisions with the obstacle. However this safer trajectory distributions entailed a decrease in the task performance (Figure 4), where more iterations were required to train the NN policy in contrast to the other two formulations. On the other hand, the training phase of the proposed D-MDGPS generated less dangerous robot end-effector movement than the standard MDGPS. Moreover, the resulting NN policy showed a performance quite similar to the standard MDGPS. Therefore, the proposed approach offers a good compromise that leads the robot to fail less

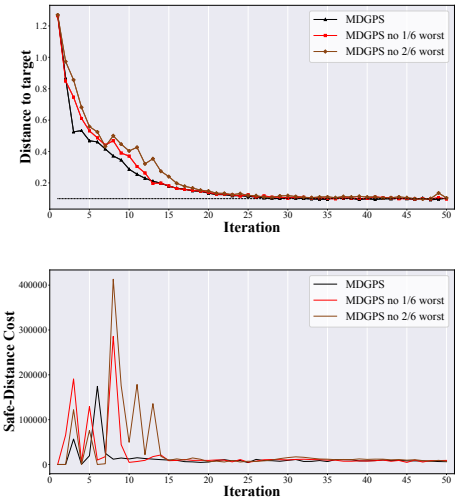

Fig. 2: Disregarding the worst samples in the SL step of MDGPS does not only avoid that the planar robot keeps failing, but also increase them. *MDGPS* label means that all the 6 samples for each local policy are considered. *MDGPS no 1/6 worst*, means that the worst sample (the one with highest cost) is disregarded. *MDGPS no 2/6 worst* means that the two samples with highest cost are disregarded.

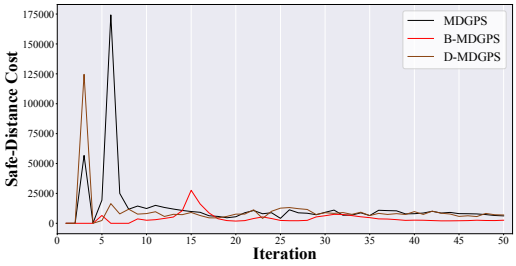

Fig. 3: Total safe-distance cost, $\sum_{t=1}^{T} \max(d_{\text{SAFE}} - d(\mathbf{o}_t), 0)$, incurred by the trajectory samples of the planar manipulator robot in each iteration.

and smoothly update its policy.

Note that there is a peak in the safe-distance cost at iteration 3 in Fig. 3. This due to the fact that the robot end-effector is initially quite far away from the target, which can be considered as an undesired behavior based on the given cost function. Consequently, D-MDGPS fits a bad trajectory distribution to these samples, and then by constraint (12), the new trajectory distributions are different than these trajectories. Despite these updates are quite aggressive and the robot generates trajectories with high safe-distance cost at iteration 3, these executions are now considered bad, and then the updated trajectory distributions do not generate such negative trajectories in the next iteration (iteration 4).

B. CENTAURO robot

1) Description: The second reaching task was carried out by a simulated CENTAURO robot. In this environment, the

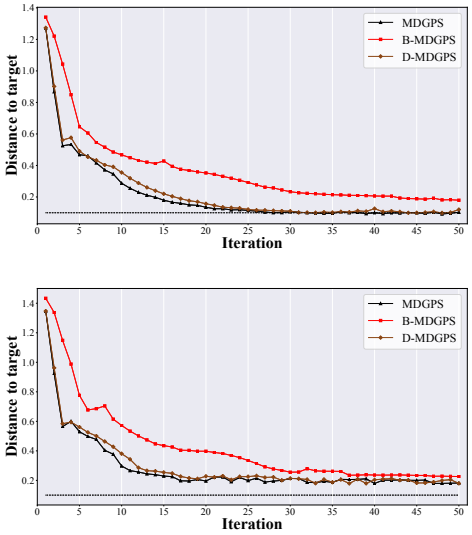

Fig. 4: Final distance from the end-effector of the planar robot to the target Cartesian pose. Top figure shows the performance on the initial conditions used to train the policy, and the bottom figure, the performance on the test set.

humanoid seeks to reach a Cartesian pose while avoiding a cylindric obstacle that is halfway, as shown in Fig. 5. The state of the task is defined by $\mathbf{x} \in \mathbb{R}^{26}$, composed of joint positions of the right arm $\mathbf{q} \in \mathbb{R}^7$, joint velocities of the right arm $\dot{\mathbf{q}} \in \mathbb{R}^7$, and both position and orientation errors between the right hand of the robot and the target $\mathbf{g} \in \mathbb{R}^6$ and the obstacle $\mathbf{o} \in \mathbb{R}^6$. The action $\mathbf{u} \in \mathbb{R}^7$ corresponds to the robot right arm task-torque commands. As we can notice, this learning task is more complex than the one in the previous experiment because, first, its state-action space has higher dimensionality, and second, the target is closer to the obstacle which increases the possibility of colliding with the obstacle.

The cost function of the task is similar than the previous experiment, except by the use of a Lorentzian ρ-function [21]. Specifically, we replace $(\gamma_1 + ||\mathbf{g}_T||^2)^{1/2}$ by $\log(\gamma_1 + ||\mathbf{g}_T||^2)$ in the final cost and $(\gamma_2 + ||\mathbf{g}_t||^2)^{1/2}$ by $\log(\gamma_2 + ||\mathbf{g}_t||^2)$ in the stage cost. With this change, we now consider $w_1 = 300$, $w_2 = 300$, $w_3 = 500$, $w_4 = 30$, $w_5 = 30$, $w_6 = 50$, $w_7 = 0.1$, $\gamma_1 = 10^{-5}$, $\gamma_2 = 10^{-5}$, and $d_{\text{SAFE}} = 0.217$.

The architecture of the NN global policy consisted of two fully connected hidden layers of 64 rectified linear units each. Similarly to the previous experiment, four TVLG controllers ($i = 4$) were iteratively optimized as guiding policies.

2) Results: Figure 6 shows the accumulated safe-distance cost generated by CENTAURO during the exploration phase of the reaching task. As in the previous experiment, both B-MDGPS and D-MDGPS generated safer trajectories because they involved lower safe distance cost. On the other hand, Fig. 7 shows the final distance of the right hand with respect to the target. As we can notice, because both B-MDGPS and D-MDGPS force the updated trajectory distribution to be dissimilar to the high safe-distance cost trajectories, the

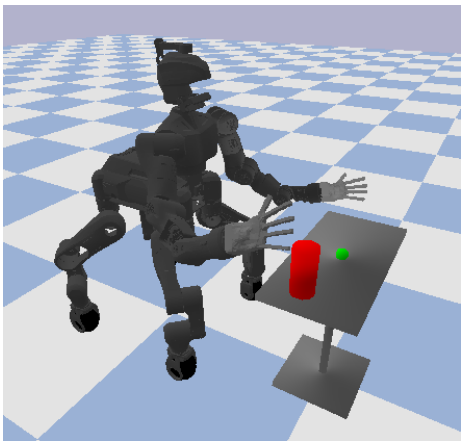

Fig. 5: Reaching task of CENTAURO. The humanoid is requested to learn to reach a desired Cartesian pose (depicted in green) with its right arm without touching the obstacle (red cylinder).

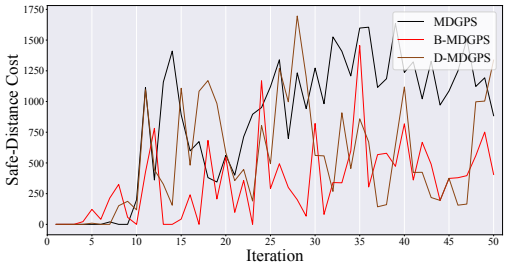

Fig. 6: Total safe-distance cost, $\sum_{t=1}^{T} \max(d_{\text{SAFE}} - d(\mathbf{o}_t), 0)$, of the trajectory samples generated by the CENTAURO robot in each iteration.

new trajectories generated at the next iteration have lower safe-distance cost. For this reason, we can see strong changes at the following iterations. Unfortunately, because the target is designed to be closer to the cylinder, in comparison to the previous experiment, the proposed method tries to minimize the total cost function, and then tries again to generate trajectories closer to the desired Cartesian pose, and then because the stochasticity of the local policies, close again to the obstacle. This situation generated the behavior observed in the Fig. 7.

VI. DISCUSSION AND FUTURE WORK

We presented a model-based trajectory-centric RL algorithm that explicitly considers good and bad experiences by bounding the KL divergence between the new trajectory and trajectory distributions encoding successful and failed executions. Good and bad trajectories are obtained from samples with low and high costs, respectively. However, this approach can be easily extended to include human assessments regarding the goodness of every robot execution, so that the trajectory distributions are influenced by human expertise.

The proposed formulation allows the robot to reduce the cost related to failures during the training phase, resulting

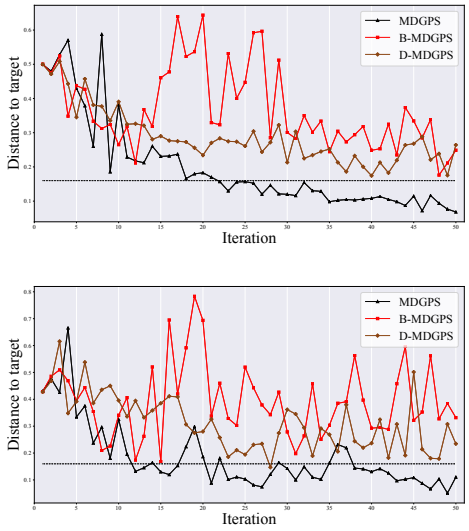

Fig. 7: Final distance from the CENTAURO's right-hand to the target Cartesian pose. Top figure shows the performance on the initial conditions used to train the policy, and the bottom figure, the performance on the test set.

in safer explorations when compared to approaches that do not explicitly consider dualist constraints. We additionally proposed to use the trajectories optimized with this method as guiding samples in a GPS setting. Specifically, we adapted the MDGPS formulation to train a high-dimensional global policy represented by a general-purpose neural network. In this way, we obtained a deep neural network policy that considered both good and bad behaviors in its learning process, which extends the MDGPS learning capabilities.

We evaluated our approach in two reaching tasks using a simulated planar robot and a simulated humanoid robot. The reported results supported our hypothesis that specifying dualist updates reduces the cost related to failures during the exploration phase. Additionally, we showed that discarding bad samples from the training dataset in which the global policy is trained on, does not necessarily avoid that the policy generates bad trajectories.

A possible limitation of our approach is that the proposed optimization could be infeasible because there is no trajectory distribution satisfying all the hard constraints at the same time. Additionally, considering dualist constraints might produce very conservative or aggressive policy updates, then increasing the number of samples required to converge. A possible solution to overcome these problems is to automatically adjust the upper and lower bounds of the good and bad trajectories at each iteration, similarly as done with the bound between the new policy and the TVLG controller approximation of the global policy in MDGPS.

The dualist GPS problem formulated in this paper requires that different sets of good and bad trajectories for each initial condition are defined. It may be interesting to find a method that permits to generate these sets from GPS formulations that do not require to deterministically reset into initial states [22]. Finally, the dualist constraints of our algorithm operate at the level of the induced trajectory distributions $p(\tau)$. Our future work will consider to impose these constraints at each time step in order to modify each linear-Gaussian policy with respect to their expected Q-value, similarly to [23]. With this change, we expect that the dualist constraints have a different effect in specific regions of the resulting trajectories.

References

[1] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics." *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization." in *International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies." *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 39, pp. 1–40, 2016.

[5] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," in *Proceedings of Robotics: Science and Systems (RSS)*, 2014.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey." *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] W. Montgomery and S. Levine, "Guided policy search as approximate mirror descent," in *NIPS*, 2016.

[8] A. Colomé and C. Torras, "Dual REPS: A generalization of relative entropy policy search exploiting bad experiences," *IEEE Transactions on Robotics*, 2017.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference*, 2005, pp. 300–306.

[11] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IROS*, 2012, pp. 4906–4913.

[12] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics." in *NIPS*, 2014.

[13] R. Lioutikov, A. Paraschos, J. Peters, and G. Neumann, "Sample-based informationl-theoretic stochastic optimal control," in *ICRA*, 2014, pp. 3896–3902.

[14] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *ICRA*, 2017, pp. 3381–3388.

[15] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 331–342, 2012.

[16] A. Rai, G. De Chambrier, and A. Billard, "Learning from failed demonstrations in unreliable systems," in *Humanoids*, 2013, pp. 410–416.

[17] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *AAMAS*, 2016, pp. 1060–1068.

[18] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstration using leveraged gaussian processes and sparse-constrained optimization," in *ICRA*, 2016, pp. 470–475.

[19] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *AAAI Conference on Artificial Intelligence*, 2010, pp. 1607–1612.

[20] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *ICRA*, 2016, pp. 528–535.

[21] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search." in *ICRA*, 2015, pp. 156–163.

[22] W. Montgomery, A. Ajay, C. Finn, P. Abbeel, and S. Levine, "Reset-free guided policy search: efficient deep reinforcement learning with stochastic initial states," in *ICRA*, 2017, pp. 3373–3380.

[23] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2017.