Learning Robust Manipulation Skills with Guided Policy Search via Generative Motor Reflexes

Philipp Ennen¹, Pia Bresenitz¹, Rene Vossen¹, Frank Hees¹

Abstract—Guided Policy Search enables robots to learn control policies for complex manipulation tasks efficiently. Therein, the control policies are represented as high-dimensional neural networks which derive robot actions based on states. However, due to the small number of real-world trajectory samples in Guided Policy Search, the resulting neural networks are only robust in the neighbourhood of the trajectory distribution explored by real-world interactions. In this paper, we present a new policy representation called Generative Motor Reflexes, which is able to generate robust actions over a broader state space compared to previous methods. In contrast to prior state-action policies, Generative Motor Reflexes map states to parameters for a state-dependent motor reflex, which is then used to derive actions. Robustness is achieved by generating similar motor reflexes for arbitrary states. We evaluate the presented method in simulated and real-world manipulation tasks, including contact-rich peg-in-hole tasks. Using this evaluation tasks, we show that policies represented as Generative Motor Reflexes lead to robust manipulation skills also outside the explored trajectory distribution with less training needs compared to previous methods. Therefore, the presented approach serves as a step towards reliable applications of reinforcement learning for manipulation.

I. INTRODUCTION

Guided Policy Search (GPS) is a model-based reinforcement learning approach that allows for learning manipulation skills in a continuous state and action space [1]. In reinforcement learning, manipulation skills are represented by policies which equal a transfer function to compute state-dependent actions. Guided Policy Search is known as a sample-efficient approach which learns a neural network policy in two phases. First, a trajectory-centric model-based reinforcement learning is used to train local trajectories for different start and goal conditions. Then, these local trajectories serve as training data for a supervised learning of a neural network policy. An essential requirement for many applications in robotics is a predictable and reliable behaviour of policies. For this reason, the policies need to be robust against unmodeled and unexpected disturbances in the state space. However, in GPS the resulting neural network policies are only robust in a small area around the explored state space since the training data for the neural network only contains a moderate amount of trajectories. In case of unexpected state disturbances during test time, a policy easily produces unstable robot behaviour. For that reason, our work aims to reach robustness, even outside the distribution of the explored trajectories, cf. Figure 1.

We introduce a novel and robust policy representation called Generative Motor Reflexes (GMR), cf. Figure 2. This policy works as a two-step approach deriving an action. In the first step, a neural network predicts parameters of a motor reflex for a given state. In the second step, the parametrized motor reflex is used to derive stabilizing actions. A motor reflex is a simple motion pattern with state feedback that shares the same parameters for multiple states. Generative Motor Reflexes increase the robustness if they fulfill two criteria: (i) the generated motor reflex parameters are drawn from the distribution of the explored trajectories and (ii) the generated motor reflex has stabilizing properties. Afterwards, during test time the stabilizing properties of a motor reflex retain for arbitrary states. Compared to prior state-action policies, our approach results in a constrained action space by the motor reflex and as a consequence state disturbances are not yielding a self-resonating error in action space. To summarize, our main contributions are:

1) A novel neural network policy representation called Generative Motor Reflex that uses motor reflexes as an intermediate step for deriving stabilizing actions and
2) an adapted version of Guided Policy Search, which allows for training a Generative Motor Reflex policy.

We derive our approach by reviewing the related work in the field of reinforcement learning and in particular robust
reinforcement learning. Then, we explain our Generative Motor Reflex policy in detail and present a training algorithm based on Guided Policy Search. Finally, we evaluate our approach on simulated and real-world experiments in the continuous control domain.

II. RELATED WORK

Reinforcement learning has been applied successfully in simulated and real-world robotic manipulation [3], [4], [5], [8], [9], [10], locomotion [6], [2] and autonomous vehicles [11]. Many of the demonstrated scenarios used tailored policy representations or discretized action spaces. Due to the difficulties in training continuous high-dimensional policies efficiently, the parameter space often contains less than 100 parameters [7]. Although notable research has been conducted in learning methods for high-dimensional neural network policies, the methods only been developed to the point where they could be applied in the continuous control domain with moderate state and action spaces, which is typically around seven degrees-of-freedom [12]. Neural networks often lack robustness, so that today’s applications are limited, in particular in continuously controlled systems. Thus, recent research addresses robustness in two ways: robust policy representations and robust reinforcement learning, which are both reviewed in the following.

A. Robust Policy Representations

One approach for robust policies is to use attractor systems as a policy representation. Dynamic Movement Primitives (DMP) belong to this class, wherein robustness is obtained by a nonlinear, error-minimizing dynamic system [13]. Dynamic Movement Primitives are used to generate linear variations of an original movement, whereby the shape of the movement is represented by a mixture of Gaussian basis functions. Although DMPs have demonstrated to be successful in many applications, their generalization ability is limited to linearly scaled variations of the original movement. Aside from attractor systems, many different general-purpose policy representations have been developed. Probabilistic Movement Primitives (ProMP), Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM) are approaches that are modeling a trajectory distribution from stochastic movements [14], [15], [16]. In recent work, Artificial Neural Networks (ANN) became very popular for their generalization abilities [17]. But these representations do not provide robustness by design. Typically, robustness of such general-purpose representations, i.e. ProMP, GMM, HMM and ANN, is only achieved, if the state disturbances are inside the trajectory distribution explored by the robot during training time. This motivated the development of several robust reinforcement learning methods.

B. Robust Reinforcement Learning

In Robust Reinforcement Learning (RRL), robustness is obtained by explicitly taking into account input disturbances and modeling errors [18]. In order to learn a robust policy, a control agent is trained in the presence of a disturbance agent, who tries to generate the worst possible disturbance. Robust Adversarial Reinforcement Learning (RARL) uses an adversarial neural network for the disturbance agent [19]. The adversarial is trained jointly with the control agent and learns an optimal destabilization policy.

A related approach is Training with Adversarial Attacks (TAA) [20]. Instead of training an adversarial, knowledge about the Q-value is exploited to craft optimal adversarial attacks. The Q-value describes the expected cost-to-go of an agent given a state and an action. In TAA, adversarial attacks are seen as disturbances that result from the worst possible action, which corresponds to the smallest Q-value in a state. However, in RRL robustness is reached by exploring those states, which could potentially lead to unstable behaviour. In contrast, GPS trains a policy by imitating multiple trajectory distributions learned by local, trajectory-centric reinforcement learning for different start and goal conditions [1]. As a result, neural network policies are only robust in the neighbourhood of the explored trajectory distributions. The extension of GPS for RRL would require a more widely exploration decreasing the sample-efficiency.

In the following we present Generative Motor Reflexes that are robust in a much broader state space compared to prior neural network policies without further exploration needs.
III. GENERATIVE MOTOR REFLEX

A. Preliminaries in Guided Policy Search

In Guided Policy Search, neural network policies $\pi(u|x)$ distributed over actions $u$ and conditioned on the state $x \in \mathcal{X}$ are trained by a trajectory-centric model-based reinforcement learning method combined with an alternating optimization procedure, cf. Algorithm 1 (C- and S-step). First, the robot samples a mini-batch $T = \{\tau_i\}$ of trajectories $\tau_i = [x_0, u_0, \ldots, x_T, u_T]^T$, for multiple start and goal conditions $i$ and fits these trajectories to a time-varying Gaussian dynamic model $p_i(x_{t+1}|x_t, u_t)$. This Gaussian dynamic model is represented by

$$p(x_{t+1}|x_t, u_t) = \mathcal{N}(f_{\text{mix}}[x_t; u_t]^T + f_{\text{et}}, F_t)$$

with the fitted time-varying matrix $f_{\text{mix}}$, the vector $f_{\text{et}}$ and the covariance $F_t$, where the subscripts denote differentiation with respect to the vector $[x_t; u_t]$. Now, the goal of the C-step is to compute improved local policies $p_i(u_t|x_t)$ by minimizing a quadratic cost function

$$J(\tau) = \sum_{t=0}^{T} E_{p_i(\tau)}[l(x_t, u_t)]$$

of the form

$$l(x_t, u_t) = \frac{1}{2}[x_t; u_t]^T l_{\text{mix,et}}[x_t; u_t] + [x_t; u_t]^T l_{\text{et,et}} + \text{const}$$

while the change of the local policies is bounded by a trust-region constraint. The constraint used in GPS is the Kullback-Leibler (KL) divergence to the previous policy distribution $D_{\text{KL}}(p_i(\tau)||\pi(\tau)) \leq \epsilon$, where the policy distribution is computed by

$$p_i(\tau) = p_i(x_0) \prod_{t=0}^{T-1} p_i(x_{t+1}|x_t, u_t) p_i(u_t|x_t).$$

Afterwards, the weights of the neural network policy $\pi$ are trained to imitate the trajectory distribution $p_i(\tau)$ using supervised learning (S-step).

B. Problem Formulation

In Guided Policy Search the robot explores a state distribution $p(\mathcal{X}_{\text{train}})$, which is a subset of the state space $\mathcal{X}$ [1]. In this explored state distribution $p(\mathcal{X}_{\text{train}})$, policies trained by Guided Policy Search are typically robust against state noise. However, robustness outside of $p(\mathcal{X}_{\text{train}})$ is usually not obtained. Therefore, we aim to learn a stochastic policy

$$u \sim \pi(u|x + \epsilon)$$

that is robust against state noise $\epsilon$ in a broader state space than $p(\mathcal{X}_{\text{train}})$.

For this purpose, the Generative Motor Reflex policy calculates the action via a compressed state space $z$, generated motor reflex parameters $\Psi$ and a state-dependent motor reflex $\pi_{\Psi}(u|x)$ (cf. Figure 3). In this policy, the compressed state $z$ is determined by a neural network using weights $\Theta_z$ and the motor reflex parameters $\Psi$ are determined by a neural network using weights $\Theta_{\Psi}$. The motor reflex parameters $\Psi = [\Psi_K; \Psi_K; \Psi_{\Sigma}]$ form a motor reflex $\pi_{\Psi}(u|x)$, which is a stochastic local motion pattern that computes actions using the framework of linear Gaussian controllers.

$$u \sim \mathcal{N}(\Psi_K x + \Psi_K; \Psi_{\Sigma})$$

Linear Gaussian controllers yield linear stabilizing properties which, however, current GPS methods do not explicitly consider so far.

During training time, the robot explores a compressed state distribution $p(\mathcal{Z}_{\text{train}})$ in tandem with the state distribution $p(\mathcal{X}_{\text{train}})$. Robust action inference is reached, if during test time arbitrary states $x$ lead to a compressed state $z$, which is inside (or close) to the distribution $p(\mathcal{Z}_{\text{train}})$. Then, the stabilizing properties of linear Gaussian controllers are explicitly represented by our policy.

This leads to a policy behaviour, where inside $p(\mathcal{X}_{\text{train}})$, a GMR keeps the generalization ability of neural networks and outside of $p(\mathcal{X}_{\text{train}})$, the linear stabilizing properties of linear Gaussian controllers are still being utilized.

C. Generative Motor Reflex Policy

In order to enable these stabilizing properties, the compressed state space $\mathcal{Z}$ must keep the trained state space $\mathcal{X}_{\text{train}}$ sufficiently diverse and at the same time as compressed as possible. For this purpose, GMR exploits the idea of a variational autoencoder (VAE) [21]. A VAE consists of two
concatenated networks: an encoder network, which maps an input to a latent representation and a decoder network, which retrieves an approximation of the original input from the latent representation (cf. Figure 3).

In GMR, the VAE enables to compress the robotic state $x$ to a latent representation $z$ while $z$ is sufficiently diverse to reconstruct the robotic state $x$ with the decoder network. By introducing noise in $z$ during the S-step, the latent representation does not only represent the exact state $x$ but also similar states. This increases the robustness of the reconstruction of the original state and as a consequence the robustness of the generated motor reflex parameters.

For retrieving $z$, a mean $\mu_z$ and a variance $\sigma_z$ is determined first using an encoding neural network $h_{\text{enc}}$ under the weights $\Theta_z$.

$$ [\mu_z; \sigma_z] = h_{\text{enc}}(x; \Theta_z) $$

Then, during training, the latent state $z$ is retrieved by Gaussian sampling of $\mu_z$ and $\sigma_z$. This sampling step enables robustness in the estimation of $z$ and leads to a robust translation of $z$ to the motor reflex parameters $\Psi$. Note that during test time, $z$ is directly set to the mean value $\mu_z$.

$$ z \sim \mathcal{N}(\mu_z, \sigma_z \odot I) $$

Now, the state translation neural network $h_{\text{trans}}$ with weights $\Theta_q$ translates the latent representation $z$ in motor reflex parameters $\Psi = [\Psi_K; \Psi_k; \Psi_\Sigma]$.

$$ [\Psi_K; \Psi_k; \Psi_\Sigma] = h_{\text{trans}}(z; \Theta_q) $$

The parameters $\Psi$ are then used as parameters for a linear Gaussian controller (cf. Equation 1). By sampling from this controller, the action $u$ to be executed is finally retrieved.

**D. Generative Motor Reflex Training**

In Guided Policy Search, the policy weights $\Theta_x$ and $\Theta_q$ are trained by supervised learning. For that, the C-step in GPS provides a dataset $D$ which are tuples of the form $(x; \Psi) \subseteq D$. Using this dataset, a variant of gradient descent is used to optimize the weights by minimizing the loss term, denoted as $L_{\text{GMR}}$ (cf. Figure 3).

The loss term has to enforce a compressed and meaningful latent state representation while the motor reflex parameters are generated with high precision. Therefore, we propose a loss term which consists of four parts: (i) minimizing the mean-squared state reconstruction loss, (ii) the KL divergence from a unit distribution to the latent state distribution, (iii) the mean-squared prediction error of the motor reflex parameters and (iv) an L2 regularization term. Therein, the sum of (i) and (ii) is the standard loss function of a variational autoencoder, where (ii) forces the compression of the state space to a latent state space which is in VAE a unit Gaussian distribution. The mean-squared prediction error (iii) trains the translation model for generating motor reflex parameters and (iv) is a regularization term that improves overall stability and is already used by previous GPS policies.

$$ L_{\text{GMR}} = \frac{1}{|D|} \sum_{x, \Psi \in D} \left(\|x - h_{\text{dec}}(z; \Theta_z)\|_2 \right) $$

$$ + \alpha D_{KL}(z||\mathcal{N}(0, I)) $$

$$ + \|\Psi - h_{\text{trans}}(z; \Theta_q)\|_2 $$

$$ + \beta \sum \Theta^2 $$

In this loss term $\beta$ weights the influence of the L2 regularization and $\alpha$ the influence of the KL divergence on the overall loss. A greater value for $\alpha$ results in a more compressed latent space, but reduces the ability to represent the state space exactly. Therefore, we propose to increase $\alpha$ linearly depending on the current $n$-iteration, which is $\alpha = \eta/n$. The increase forces the weights $\Theta_z$ to slowly minimize part (ii) of $L_{\text{GMR}}$ over all iterations $N$. This avoids a local optimal solution for the latent state representation which would not be sufficient for a precise translation to motor reflex parameters.

**E. Generative Motor Reflex with Guided Policy Search**

Algorithm 2 presents the adapted Policy Search method for training GMR, where we chose a GPS variant, which is the Mirror Descent GPS (MDGPS) [22]. Other variants of GPS can be adapted analogously.

In MDGPS, the C-step minimizes the following Lagrangian loss term:

$$ L_{\text{GPS}} = J(\pi) + \sum_{t=1}^T \lambda_t D_{\text{KL}}(p_i(\tau)||\pi(\tau)) $$

This loss term consists of two parts: (i) the trajectory loss term $J(\pi)$ and (ii) the constraint $D_{\text{KL}}(p_i(\tau)||\pi(\tau)) \leq \epsilon$, which bounds the change in the optimized trajectory distribution $p_i(\tau)$ and by utilizing the Lagrange multiplier $\lambda$. Now, the loss term $L_{\text{GPS}}$ is minimized using dual gradient descent (DGD).

DGD is an optimization approach, which enables to incorporate Lagrange multipliers. For DGD in GPS, three steps are performed: Given an initial $\lambda$, trajectory optimization is utilized for minimizing $L_{\text{GPS}}$ (C1- and C2-step). Then, the dual variable $\lambda$ is updated stepwise until the constraint

**Algorithm 2 Mirror Descent GPS with GMR**

| Step | Description |
|------|-------------|
| 1. | for $n$-iteration $n = 1$ to $N$ do |
| 2. | Generate samples $T = \{\tau_i\}$ by running $p_i$ or $\pi$ |
| 3. | Fit Gaussian dynamics $p_i(x_{t+1}|x_t, u_t)$ |
| 4. | Compute $\alpha = n/N$ |
| 5. | do |
| 6. | C1-step: LQR backward $D \leftarrow [\Psi_K; \Psi_k; \Psi_\Sigma]$ |
| 7. | C2-step: LQR forward $D \leftarrow x_i$ |
| 8. | C3-step: Adjust dual variable $\lambda$ |
| 9. | while $D_{\text{KL}}(p_i(\tau)||\pi(\tau)) - \epsilon > 0$ |
| 10. | S-step: $\pi \leftarrow \arg \min_{\Theta_x, \Theta_z, \Theta_q} L_{\text{GMR}}(D, \alpha)$ |
violation $D_{KL}(p_t(\tau)||\pi(\tau)) - \epsilon$ (C3-step) is met. In prior
GPS methods, the output of this minimization process is an
optimized trajectory distribution $p_t(\tau)$. However, for training
GMR we need a dataset $\mathcal{D}$ with pairs of states $x$ and
corresponding motor reflex parameters $\Psi$. This dataset can be
retrieved during the C1 and C2-step.

Therein, the optimized trajectory distribution is computed
by a linear-quadratic regulator (LQR). A LQR consists of
a backward (C1-step) and a forward pass (C2-step). Within
the LQR backward pass, a value function is minimized that
represents the accumulated cost starting from the state $x_t$
to goal state $x_T$.

$$V(x_t) = \min_{u_t} Q_t(x_t, u_t) = \min_{u_t} \mathcal{L}_{GPS}(x_t, u_t) + E[V(x_{t+1})]$$

With having computed an approximation of the Q-function
by a second-order Taylor expansion, the optimal action can
be found by the partial derivation with respect to $u_t$:

$$u_t = \arg\min_{u_t} Q_t(x_t, u_t) = -Q^{-1}_{u,u}(Q_{ut} + Q_{ux}x_t) \quad (3)$$

Now, Equation 3 can be transformed to a linear Gaussian
controller by substituting the following motor reflex parameters
in timestep $t$:

$$\Psi_{Kt} = -Q^{-1}_{u,u}Q_{ut}$$
$$\Psi_{kt} = -Q^{-1}_{u,u}Q_{ux}$$
$$\Psi_{\Sigma t} = Q^{-1}_{u,u}$$

As proposed in previous GPS approaches [22], we set the
motor reflex covariance $\Psi_{\Sigma t}$ to the inverse of the action
related Q-value $Q^{-1}_{u,u}$. The intuition behind this is to keep the
exploration noise $\Psi_{\Sigma t}$ low in case that the actions change the
Q-value significantly. Now, the first and second derivatives
of the value function can be derived as follows:

$$Q_{xu, xut} = l_{xu, xut} + f_{xu}V_{x, x_{t+1}}f_{xut}$$
$$Q_{xut} = l_{xu, xut} + f_{xu}V_{x, x_{t+1}}$$
$$V_{x, xt} = Q_{x, x_t} + Q_{u, x_t}\Psi_{Kt}$$
$$V_{x, xt} = Q_{x, x_t} + Q_{u, x_t}\Psi_{\Sigma t}$$

Using these equations, the LQR backward pass calculates
$\Psi_{Kt}$, $\Psi_{kt}$ and $\Psi_{\Sigma t}$ as well as the Q- and V-values for each
timestep. Then, in a following forward pass (C2-step) we
exploit the learned dynamics to retrieve the corresponding
states $x_t$ for $\Psi_{Kt}$, $\Psi_{kt}$, $\Psi_{\Sigma t}$ given an initial state $x_0$. After
the C-steps, the GMR weights $\Theta_x, \Theta_z, \Theta_\psi$ are trained within
the S-step. This minimizes the loss function presented in
Equation 2 and leads to the final GMR policy.

IV. EXPERIMENTS

The proposed learning algorithm is evaluated by using a
six axes robot arm of the type Kinova Jaco 2 for simulated
and real world manipulation tasks (cf. Figure 4). Within the
scope of this work the object to be manipulated is rigidly
connected to the gripper. The implementation is build upon
the guided policy search toolbox [23] and is utilizing Gazebo
for simulation.

We evaluate two properties of our approach: (i) the learning
performance and (ii) the robustness of GMR. The evaluation
scenarios include two conditions of simulated reaching tasks
(cf. Figure 3A, 3B) and two conditions of real-world peg-
and-hole tasks with a light press fit (cf. Figure 3C, 3D).

In each iteration, the robot collects five samples per condition
with a length of $T = 80$ timesteps. We trained the robot
each experiment for $N = 10$ iterations. This process
leads to a total of 50 trajectory samples for each condition.
The continuous state space of the robot consists of six joint
positions and six joint velocities. The action space includes
six joint torques that are controlled by the policy with 20 Hz.
This leads to a parameter space of the motor reflex with 114
dimensions ($\Psi_{K} \in \mathbb{R}^{6 \times 12}, \Psi_{k} \in \mathbb{R}^{6}, \Psi_{\Sigma} \in \mathbb{R}^{12 \times 6}$).

We set the L2 regularization term to $\beta = 0.0002$. The
weights $\Theta$ of the policy topology presented in Figure 3
are optimized by the ADAM solver, whereby the number
of training epochs was set to 400 and the batch size to 20.
We compared our policy with the prior state-action policy
trained by MDGPS. Here we used the same settings for the
policy training, except for the difference in the topology that
was set to a fully connected neural network with two hidden
layers with 64 rectified linear units. The supplementary
video compares the resulting motion behaviour of GMR and
MDGPS policies.\footnote{https://philippente.github.io/pub/GMR.html}

A. Learning Performance

First, we evaluated the learning performance of the GMR
training compared to MDGPS and the linear Gaussian
controller base layer, where only the C-step is performed
(hereinafter LQR base layer). In Figure 5 the mean squared
error (MSE) of the final state distance for the reaching task in
case of one and two conditions is compared between GMR,
MDGPS and the LQR base layer.

During training time of GMR policies, new LQR base layers
are computed after each iteration, which are then used to
continue the supervised training of the GMR policies. For
both training scenarios (cf. Figure 4), GMR requires two
iterations to imitate the LQR base layer reliably. In contrast to the expectations, the GMR solution outperforms the LQR base layer after a few iterations. This property is a result of the generalization ability of the GMR, which is learned by training the policy on two LQR base layers of successive iterations. The learning progress of GMR shows a small variance in the performance, which points out the reliability of our approach. Compared to this, the performance of the MDGPS policy has a high variance during training and usually needs 2 - 4 times more iterations for a successful imitation of the LQR base layer.

B. Robustness

Robustness to unseen states is the key advantage of GMR. We show, that GMR produces stable motions over a much broader state space compared to MDGPS. First, we trained the simulated robot on scenario A and B (cf. Figure 4). Then, we set the initial robot state to 50 random, uniform distributed configurations over the whole state space. From there, the robot executed the policy, which is either a GMR or MDGPS policy. The diagram in Figure 5 shows the resulting trajectories of the end effector for both policies. Table I provides an overview of the success rate for reaching the final state, which is defined as MSE < 0.1.

For the simulated scenarios A and B, GMR reaches the final state for all random initial states. In comparison, the MDGPS policy fails in 38 of 50 trials in case of one training condition. Within the simulated scenario, this problem can be solved by using more training conditions (which requires more training time). Nevertheless, we usually observe an extreme overshooting at the end of the motions for all scenarios, cf. Figure 5c and 5d.

However, in the real-world scenarios C and D the high failure rate remains even for two training conditions. Usually, the MDGPS policy starts to oscillate even by small state disturbances. If the state disturbance increases, the MDGPS policy diverged from the goal state, whereas the GMR policy typically reaches the goal state with less overshooting.

In conclusion, we can state that GMR policies are an approach to reach robustness outside the explored trajectory distribution even with a low number of training conditions.

V. Conclusion and Future Work

We could improve the robustness of neural network policies by using motor reflexes as an intermediate step and evaluated our approach for manipulation tasks. So far, these motor reflexes are linear Gaussian controllers. In future work, we will research also other motor reflex representations, i.e. error minimizing dynamic systems like Dynamic Movement Primitives. In addition, a promising direction for further research is the embedding of vision-based feature extractors for a more general usage of GMR.

However, in terms of the robustness of GMR, stability analysis could be researched. A key property of generative models is the ability to generate samples from a latent space, which are in GMR pairs of states and motor reflex parameters. This is not tackled in this work yet, but could potentially be an approach for validating GMR.
ACKNOWLEDGMENT

This contribution is funded by VDMA as part of the project “InPulS - Intelligent processes and learning systems” of the Forschungskuratorium Maschinenbau (FKM).

REFERENCES

[1] Levine, S. and Abbeel P. “Learning neural network policies with guided policy search under unknown dynamics.” Advances in Neural Information Processing Systems. 2014.

[2] Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J. and Cheng, G. “Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot.” The International Journal of Robotics Research 27.2 (2008): 213-228.

[3] Peters, J. and Schaal, S. “Reinforcement learning of motor skills with policy gradients.” Neural networks 21.4 (2008): 682-697.

[4] Levine, S., Wagener, N., and Abbeel, P. "Learning contact-rich manipulation skills with guided policy search." Robotics and Automation (ICRA), 2015 IEEE International Conference on (pp. 156-163).

[5] Theodorou, E., Buchli, J. and Schaal, S. "Reinforcement learning of motor skills in high dimensions: A path integral approach." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[6] Kohl, N. and Stone, P. "Policy gradient reinforcement learning for fast quadrupedal locomotion." Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on. Vol. 3. IEEE, 2004.

[7] Deisenroth, M. P., Neumann, G. and Peters, J. ”A survey on policy search for robotics.” Foundations and Trends in Robotics 2.12 (2013): 1-142.

[8] Peters, J., Mülling, K. and Altun, Y. "Relative Entropy Policy Search.” AAAI. 2010.

[9] Kalakrishnan, M., Righetti, L., Pastor, P. and Schaal, S. "Learning force control policies for compliant manipulation.” Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[10] Inoue, T., De Magistris, G., Munawar, A., Yokoya, T. and Tachibana, R. Deep reinforcement learning for high precision assembly tasks. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on (pp. 819-825).

[11] Abbeel, P., Coates, A., Quigley, M. and Ng, A. Y. An application of reinforcement learning to aerobatic helicopter flight. In Advances in neural information processing systems (2007): 1-8.

[12] Gu, S., Holly, E., Lillicrap, T. and Levine, S. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates.” Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.

[13] Schaal, S. “Dynamic movement primitives—a framework for motor control in humans and humanoid robotics.” Adaptive motion of animals and machines. Springer, Tokyo, 2006. 261-280.

[14] Paraschos, A., Daniel, C., Peters, J. R. and Neumann, G. "Probabilistic movement primitives.” In Advances in neural information processing systems (2013): 2616-2624.

[15] Calinon, S., Guenter, F. and Billard, A. On learning, representing, and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 37(2), (2007): 286-298.

[16] Billard, A., Calinon, S., Dillmann, R. and Schaal, S. Robot programming by demonstration. In Springer handbook of robotics (2008): 1371-1394. Springer, Berlin, Heidelberg.

[17] Levine, S., Finn, C., Darrell, T. and Abbeel, P. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1), (2016): 1334-1373.

[18] Morimoto, J. and Kenji D. "Robust reinforcement learning.” Neural computation 17.2 (2005): 335-359.

[19] Pinto, L., Davidson, J., Sukthankar, R. and Gupta, A. “Robust adversarial reinforcement learning.” arXiv preprint [arXiv:1703.02702] (2017).

[20] Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., Chowdhary, G. “Robust deep reinforcement learning with adversarial attacks.” In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (2018): 2040-2042. International Foundation for Autonomous Agents and Multiagent Systems.

[21] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. arXiv preprint [arXiv:1312.6114] (2013).

[22] Montgomery, W. H. and Levine, S. "Guided policy search via approximate mirror descent.” Advances in Neural Information Processing Systems. 2016.

[23] Finn, C., Zhang, M., Fu, J., Montgomery, W., Tan, X.Y., McCarthy, Z., Stadie, B., Scharff, E. and Levine, S. Guided Policy Search Code Implementation. 2016. Software available from rll.berkeley.edu/gps.