Auton Robot (2017) 41:1521–1537
DOI 10.1007/s10514-016-9613-x

Autonomous exploration of motor skills by skill babbling

René Felix Reinhart[1]

Received: 15 September 2015 / Accepted: 16 December 2016 / Published online: 28 December 2016
© Springer Science+Business Media New York 2016

Abstract Autonomous exploration of motor skills is a key capability of learning robotic systems. Learning motor skills can be formulated as inverse modeling problem, which targets at finding an inverse model that maps desired outcomes in some task space, e.g., via points of a motion, to appropriate actions, e.g., motion control policy parameters. In this paper, autonomous exploration of motor skills is achieved by incrementally learning inverse models starting from an initial demonstration. The algorithm is referred to as skill babbling, features sample-efficient learning, and scales to high-dimensional action spaces. Skill babbling extends ideas of goal-directed exploration, which organizes exploration in the space of goals. The proposed approach provides a modular framework for autonomous skill exploration by separating the learning of the inverse model from the exploration mechanism and a model of achievable targets, i.e. the workspace. The effectiveness of skill babbling is demonstrated for a range of motor tasks comprising the autonomous bootstrapping of inverse kinematics and parameterized motion primitives.

Keywords Autonomous exploration · Goal babbling · Motion primitives · Parameterized motor skills · Inverse models

✉ René Felix Reinhart
freinhart@uni-bielefeld.de

[1] Research Institute for Cognition and Robotics – CoR-Lab, Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany

1 Introduction

Motion primitives are an established paradigm to decompose complex motions into simpler building blocks. Motion primitives are often initialized by demonstrations and then refined by reinforcement learning (e.g., Kormushev et al. (2010) or see Argall et al. (2009) and Kober et al. (2013) for recent reviews). However, the adaptation of motion primitives to novel task parametrizations, e.g., via points, by re-adaptation through reinforcement learning or demanding a new demonstration from a human tutor does not scale to complex tasks and introduces time-consuming learning trials where immediate skill execution is required.

Recently, the parameterization of motion primitives has been proposed to address this issue. For instance, (Matsubara et al. 2010; Reinhart and Steil 2014) parameterized via points of primitives for reaching skills, (Kober et al. 2012) performed context-dependent meta-parameter adaptation of primitives for throwing darts and playing table tennis. Stulp et al. (2013) learned a motion primitive for grasping that is parameterized by the object pose. These approaches avoid re-adaptation of a motor skill in the exploitation phase by having a parameterized model for skill production available. To be able to do so, parameterized motor skills have to capture the intrinsic variability of a skill. This has been achieved by collecting representative motions from the distribution of task parametrizations in a learning phase (Matsubara et al. 2010; Stulp et al. 2013; Reinhart and Steil 2014). These motions are then used to train a parameterized model for skill production, e.g., by means of regression techniques. Such parameterized skills can be considered as inverse models, because they map desired outcomes, e.g., traverse a via point at a certain point in time, to actions, e.g., a particular set of motion primitive parameters that achieve the desired outcome. In this context, the forward model corresponds to the process of motion gen-

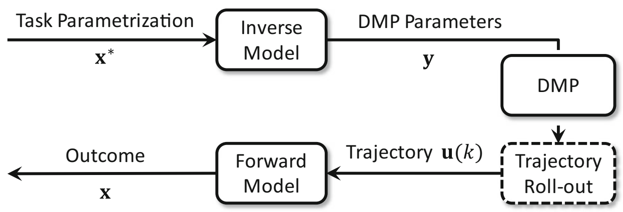


Fig. 1 A parameterized skill comprises an inverse model that maps task parameters **x**, e.g., a desired via point, to appropriate parameters of a motion primitive model **y**, e.g., Dynamic Movement Primitive (DMP, Ijspeert et al. 2013)

eration and evaluation, e.g., measuring the actual via point of the produced trajectory at a certain point in time. A sketch of such a parameterized skill using a Dynamic Movement Primitive (DMP, Ijspeert et al. 2013) is shown in Fig. 1.

While the collection of representative motions for the learning of parameterized skills has been accomplished by following the programming-by-demonstration paradigm (Matsubara et al. 2010; Stulp et al. 2013) or by optimizing a set of policies (da Silva et al. 2012, 2014a; Pontón et al. 2014), there are multiple restrictions and pitfalls connected to the acquisition of inverse models from example data. First of all, motor skills display a high degree of redundancy. For instance, there is an infinite set of motions which go through the same via point. Standard regression techniques average over ambiguous training examples, which results in invalid models if the regression problem has non-convex solution sets (Jordan and Rumelhart 1992). Second, the compilation of a suitable set of motions for inverse model learning requires careful collection of training examples, which can not be expected from non-expert users, and is time-consuming. The exhaustive exploration of the policy parameter space is neither an option because of the curse of dimensionality. Therefore, the autonomous and sample-efficient exploration of policies is essential to scale learning of parameterized motor skills.

This paper addresses these challenges within a novel bootstrapping framework referred to as *skill babbling*, which learns direct inverse models through local exploration. Inverse model learning in skill babbling starts from an initial demonstration and proceeds by incrementally learning new samples that are collected according to an exploration strategy. It extends state-of-the-art methods for efficient exploration of direct inverse models by goal babbling (Rolf et al. 2010; Baranes and Oudeyer 2013). In skill babbling, learning of direct inverse models is formulated as weighted regression problem which can be efficiently solved sequentially. A particular redundancy resolution to the inverse problem is selected by an initial demonstration and preserved during exploration of new target-action pairs by means of a weighted error criterion. Exploration is organized in task space, which is typically much lower-dimensional than the space parameterizing the actions. Due to the restriction of the inverse model to a single redundancy solution together with exploration in task space, skill acquisition is sample-efficient and scales to motor skills with high-dimensional action parametrizations. Already explored regions in task space are consolidated in a workspace model. The consolidation of a workspace model in an episodic manner without a predefined set of targets and the modular weighting scheme that enables the integration of additional task constraints contribute to the state-of-the-art implementations of goal babbling, e.g., in Rolf et al. (2010), Baranes and Oudeyer (2013). The proposed skill babbling algorithm is demonstrated in a diverse set of scenarios, including the bootstrapping of inverse kinematics as well as the exploration of parameterized motion primitives for task-specific motion generation.

The paper is structured as follows: In Sect. 2, related work is discussed. The skill babbling approach is introduced in Sect. 3. Properties of the approach are systematically analyzed in Sect. 4. The autonomous exploration of motion primitives is demonstrated in Sect. 5 for a set of scenarios. Sect. 6 concludes the paper and gives an outlook on future work.

2 Related work on parameterized skills and goal babbling

Recently, the parameterization of motion primitives has received particular attention in robotics. The first approach to systematically query parameters of Dynamic Movement Primitives (DMPs, see Ijspeert et al. (2013) for a recent review) from a compact task description by means of non-linear function approximation has been introduced by Ude et al. (2007) to the best of our knowledge. Related work by Stulp et al. (2013) studied how to compactly represent parameterized DMPs. Calinon et al. (2013) consider the automated selection of a suitable task parametrizations from candidate reference frames. The parametrization of skills according to situational context, e.g., how a ball approaches a racket, has been accomplished by learning mappings from context variables to meta-parameters of the motion primitive model (Kober et al. 2012; Kupcsik et al. 2013). Another approach addresses the task-specific Mixture of Motion Primitives (MoMP, Mülling et al. 2013) by a gating network, which is initially trained from supervised data and then refined by a reinforcement learning algorithm.

Essentially, the approaches by Ude et al. (2007), da Silva et al. (2012), Matsubara et al. (2010), Mülling et al. (2013) and Stulp et al. (2013) augment a motion primitive model with a mapping that provides mixture coefficients or model parameters in response to a task-specific input. In da Silva et al. (2012), the idea of parameterized skills is integrated into a more general framework, which in addition includes the unsupervised discovery of multiple sub-manifolds that together define a skill. In these previous works, skills are parametrized by their terminal state, e.g., the final position of the end effector, the target for throwing a dart, or the context. The parametrization of motion shapes has been presented by Matsubara et al. (2010), Reinhart and Steil (2014) and Reinhart and Steil (2015).

In the work referenced above, the basis for parameterized skill learning is a training set of representative motions. These training examples either stem from manual demonstration or automated optimization. In the former case, it is difficult to design the user interaction in a way such that the obtained demonstrations cover the relevant variance of the skill and that the training examples are not ambiguous. Ambiguous training samples lead to invalid learning of the inverse model in case of non-convex solutions (Jordan and Rumelhart 1992). In the latter case, a dedicated optimization process is conducted for each task parameterization in order to obtain pairs of task parameters and motion primitive parameters for training. Regularization of the optimization process, e.g., by introducing terms in the reward function that favor smooth motions, is essential to obtain a set of training examples which likely does not comprise ambiguous examples and can be learned by means of function approximation. Also in this case, there is no build-in mechanism to assure the validity of the assumption that the obtained training set can be approximated by a functional relationship. Moreover, the workspace, from which targets for optimization are selected, is typically unknown a priori. This complication further diminishes the generality of this approach for the acquisition of parameterized motion primitives.

The active learning of parameterized motion primitives has been addressed by da Silva et al. (2014b). The work in da Silva et al. (2014b) extends the previous framework in da Silva et al. (2012, 2014a) by a mechanism to select a new target, which is expected to maximize the skill performance improvement. While this framework introduces an active target selection mechanism, the training samples are still obtained by a full policy optimization stage. Therefore, this approach suffers from the same difficulties to compile a suitable training set without ambiguous samples for inverse model learning described above.

In developmental robotics, another line of research considers the autonomous exploration of motor skills, e.g. inverse kinematics (Rolf et al. 2010; Baranes and Oudeyer 2013). Rolf et al introduced the concept of goal babbling, which provides a coherent framework for bootstrapping inverse models and is inspired by infant development. Its main contribution is the introduction of a goal-directed exploration strategy. Instead of organizing exploration in the action space, which is typically high-dimensional, goal babbling iteratively learns an inverse model by trying to accomplish targets in task space. There is exploration in action space (joint angle space in Rolf et al. (2010)), however, only close to a single redundancy solution to the inverse problem. Hence, the exhaustive sampling of the action space, i.e. motor babbling, is avoided. Goal babbling scales to inverse modeling problems with more than 50 degrees of freedom (Rolf et al. 2011). The main difference of autonomous exploration by goal babbling compared to the approaches discussed above is that goal babbling does explicitly integrate the discovery and learning of inverse models instead of separating the training data collection from the inverse model learning process.

While goal babbling is in general applicable for any motor coordination task (Rolf et al. 2010), it requires a particular exploration strategy along paths in task space to resolve the non-convexity problem for the learning. This path generation is suitable for learning inverse kinematics as demonstrated for several robot platforms including a soft robot trunk (Rolf and Steil 2014). However, this path generation is rather inefficient for learning parameterized motion primitives, because then each step along those paths in task space corresponds to the execution of a complete motion.

In contrast to goal babbling, exploration is not organized along paths in skill babbling, but in an episodic manner. This is beneficial for the bootstrapping of parameterized motion primitives. Skill babbling also targets at the discovery of a single solution to the inverse problem by not exhaustively exploring the space of actions or policy parameters. It therefore inherits the sample-efficiency from goal babbling. However, there is no dedicated model of achievable targets, i.e. the workspace, in goal babbling. Skill babbling extends goal babbling by incorporating an explicit model of the workspace, a more flexible exploration scheme, and the integration of additional constraints into the inverse model learning process.

3 Skill babbling

For many tasks, learning of a motor skill can be formulated as inverse modeling problem. The idea of skill babbling is to acquire an inverse model $\mathbf{y} = g(\mathbf{x})$ from an initial target-action pair by means of autonomous exploration and learning. Actions, e.g. joint angles, are denoted by $\mathbf{y} \in \mathbb{R}^N$ and outcomes in some task space, e.g. end effector postures, by $\mathbf{x} \in \mathbb{R}^M$. Also more indirect representations of motor actions are considered in this paper. For instance, a sequence of actions can be encoded by policy parameters. To simplify the notation, the terms action and action space are used the in following, while these can take the form of policy parameters and policy parameter space, respectively. The forward model $\mathbf{x} = f(\mathbf{y})$ represents the causal relation between actions $\mathbf{y}$ and effects $\mathbf{x}$. For instance, forward kinematics in case of joint angle actions and resulting end effector postures. But also more indirect causal relations e.g., between policy parametrizations (actions) and effects (e.g. via points) can be considered as forward model. It is assumed that the outcome of an action can be observed, or an internal forward model $f(\mathbf{y})$ is available. In this paper, redundant systems with $N > M$ are considered.

Let the inverse model $\mathbf{y} = g(\mathbf{x}, \theta)$ be a function with parameters θ. The goal of the learning is to find parameters θ for the inverse model g, such that $f(g(\mathbf{x}, \theta)) \approx \mathbf{x} \, \forall \mathbf{x} \in X^\star$ in the workspace $X^\star = \{\mathbf{x} \mid \exists \, \mathbf{y} : f(\mathbf{y}) = \mathbf{x}\}$.

Besides the forward and inverse model, the skill babbling framework is composed of four further components (compare Fig. 2):

– *Exploration* For the autonomous bootstrapping of an inverse model, a mechanism to organize exploration both in task space and in action space is required. Exploration is organized in episodes, where in each episode the inverse model is probed and trained around a local region in task space, which is denoted as task seed $\mathbf{x}^{\text{seed}}$ in the following.
– *Workspace model* A representation of the already explored task space is required in order to effectively guide exploration. The workspace model W is equipped with a selection mechanism to provide a new task seed $\mathbf{x}^{\text{seed}}$ in the next exploration episode. Note that this model does in the beginning of skill babbling not yet cover the actual workspace. During the learning process, the workspace model W grows and eventually converges to the actual workspace for the considered task ($W \subseteq X^\star$).
– *Weighting scheme* For successful learning of inverse models in the presence of redundancies, it is essential to discard redundant solutions, i.e. when distinct actions $\mathbf{y}_1 \neq \mathbf{y}_2$ map to the same outcome $f(\mathbf{y}_1) = f(\mathbf{y}_2)$. To address the issue of multiple solutions to the inverse problem, at least a basic weighting scheme is required which computes weights for the supervised update of the inverse model. Additional constraints, e.g. stability of a gait for locomotion, can optionally be integrated into the weighting scheme.
– *Trajectory rollout* (optional) Note that the inverse model can output parameters of complex models, e.g., $\mathbf{y}$ can be parameters of a motion primitive, which after execution indirectly result in outcomes $\mathbf{x}$ in task space, e.g., via points of the motion generated by that motion primitive. In this case, the actions $\mathbf{y}$ have to be unrolled, e.g., in time, in order to evaluate the forward model on the outcome of the action. For notational simplicity, we write $f(\mathbf{y})$ also in this case and do not explicitly denote the trajectory roll-out.

In the following, each step of the skill babbling algorithm is described in more detail.

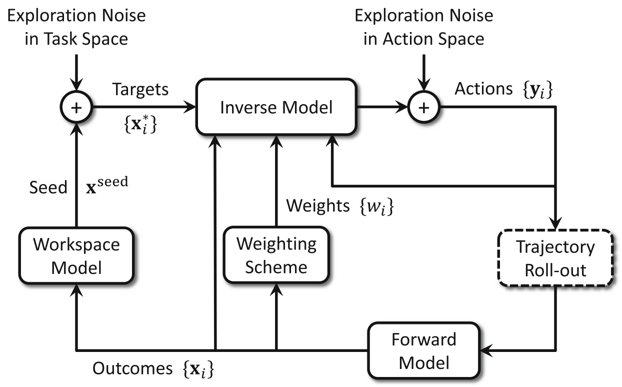

Exploration Noise in Task Space
Exploration Noise in Action Space
Targets $\{\mathbf{x}_i^*\}$
Inverse Model
Actions $\{\mathbf{y}_i\}$
Seed $\mathbf{x}^{\text{seed}}$
Weights $\{w_i\}$
Workspace Model
Weighting Scheme
Trajectory Roll-out
Forward Model
Outcomes $\{\mathbf{x}_i\}$

Fig. 2 Sketch of the skill babbling algorithm: exploration is organized in episodes and triggered by the workspace model, which provides a seed $\mathbf{x}^{\text{seed}}$ for local exploration. In each episode, the inverse model estimates actions $\{\mathbf{y}_i\}$, which are perturbed by exploratory noise in action space, to achieve the targets $\{\mathbf{x}_i^\star\}$. The actual outcomes $\{\mathbf{x}_i\}$ are observed through the forward model. Then, the inverse model is updated according to the outcomes $\{\mathbf{x}_i\}$, weights $\{w_i\}$ and actions $\{\mathbf{y}_i\}$. The outcomes $\{\mathbf{x}_i\}$ are integrated in the workspace model. Depending on the action representation, additional trajectory roll-outs are computed for actions $\mathbf{y}_i$

3.1 Exploration

Exploration starts from an initially provided action $\mathbf{y}^{\text{ini}}$. By application of the forward function, the corresponding outcome $\mathbf{x}^{\text{ini}} = f(\mathbf{y}^{\text{ini}})$ in task space can be observed. The target-action pair $(\mathbf{x}^{\text{ini}}, \mathbf{y}^{\text{ini}})$ serves as initial condition for skill babbling.

Then, exploration is organized in episodes, where in each episode R targets $\mathbf{x}_i^\star, i = 1, \ldots, R$ are drawn from a local target distribution $P(\mathbf{x})$ anchored at the current exploration seed $\mathbf{x}^{\text{seed}}$. In this paper, targets

$$\mathbf{x}_i^\star \sim \mathcal{N}\left(\mathbf{x}^{\text{seed}}, \sigma_{\text{tsk}}\right) \quad i = 1, \ldots, R \tag{1}$$

are drawn from an isotropic Gaussian distribution with mean value at the current exploration seed $\mathbf{x}^{\text{seed}}$ and variance σ_{tsk} (see Fig. 2, top left). The selection of exploration seeds $\mathbf{x}^{\text{seed}}$ can be used to bias exploration, e.g., to extend the inverse model to unexplored regions of the workspace. In principle typical criteria from active learning can be applied for the selection of $\mathbf{x}^{\text{seed}}$. In this paper, a simple selection mechanism for $\mathbf{x}^{\text{seed}}$ is applied, which is described later on in Sect. 3.2.2.

The targets $\mathbf{x}_i^\star$ are mapped to actions $\mathbf{y}_i$ via the current inverse model and corrupted with exploratory noise in action space:

$$\mathbf{y}_i = g(\mathbf{x}_i^\star) + \epsilon, \quad i = 1, \ldots, R. \tag{2}$$

The exploratory noise ϵ in action space is a zero-mean, isotropic Gaussian distribution with variance σ_{act} if not stated otherwise. The actual outcomes $\mathbf{x}_i = f(\mathbf{y}_i)$ of the corrupted actions are observed in task space after execution with the help of the forward model. Hence, a set of R triples with targets $\mathbf{x}_i^\star$, achieved outcomes $\mathbf{x}_i$ and corresponding actions $\mathbf{y}_i$ is available in each episode.

3.2 Adaptation

Subsequently to the exploration step, the inverse model and the workspace model are updated.

3.2.1 Learning of the inverse model

For supervised learning of the inverse model, the achieved outcomes $\mathbf{x}_i$ with the executed actions $\mathbf{y}_i$ serve as new training data. To resolve the problem of ambiguous training examples [non-convexity problem (Jordan and Rumelhart 1992)] for the learning of a direct inverse model $\mathbf{y} = g(\mathbf{x})$, the training samples are weighted according to a weighting scheme. Similar to goal babbling, an efficiency criterion serves as basis for the weighting, which rates actions higher that achieve a larger change in task space with smaller changes of the action variables:

$$\tilde{w}_i = \frac{||\mathbf{x}_i - \mathbf{x}^{\text{seed}}||}{||\mathbf{y}_i - \mathbf{y}^{\text{seed}}||} \quad \text{with } \mathbf{y}^{\text{seed}} = g\left(\mathbf{x}^{\text{seed}}\right). \tag{3}$$

The weights are normalized by the maximally applied action change with respect to the seed and the maximally achieved change in task space per episode:

$$w_i = \tilde{w}_i \cdot \frac{\max_{i=1,\ldots,R}\left(||\mathbf{y}_i - \mathbf{y}^{\text{seed}}||\right)}{\max_{i=1,\ldots,R}\left(||\mathbf{x}_i - \mathbf{x}^{\text{seed}}||\right)}. \tag{4}$$

Note that the weighting scheme can be flexibly extended beyond the basic efficiency criterion (4) depending on the task at hand. Assume that there are $s = 1, \ldots, S$ criteria for which weights $w_i^s \geq 0$ are computed. Then, the effective weights can be combined by

$$w_i = \prod_{s=1}^{S} w_i^s. \tag{5}$$

For learning of the inverse model g, any supervised learning algorithm that can minimize the weighted sum-of-squares error

$$E_{\text{WSOS}}(\theta) = \frac{1}{2}\sum_{i=1}^{E\cdot R} w_i ||\mathbf{y}_i - g(\mathbf{x}_i, \theta)||^2 \tag{6}$$

is applicable. Note that the sum in (6) runs over all E episodes, even though the data is only sequentially available after each episode. Therefore, sequential learning algorithms with well-behaved extrapolation are required for skill babbling. If the inverse model $g(\mathbf{x}, \theta)$ is linear in the parameters θ, online variants of weighted linear regression schemes are sufficient to implement the model update. In this paper, we focus on two such learners that are described in Appendix.

3.2.2 Learning of the workspace model

During exploration, a model W of the already explored workspace $X^\star$ is learned. The workspace model W can be implemented by density estimators, e.g., like Gaussian Mixture Models, tree-structures, or by vector quantization algorithms.

In this work, we apply a simple prototype-based vector quantization algorithm similar to Reinhart and Steil (2015). The algorithm represents the explored workspace by means of prototypical hyper-spheres with centers $\{\mathbf{c}_i\}$ and identical radii γ. A new hyper-sphere $\mathbf{c}^{\text{new}} \equiv \mathbf{x}$ is added to the workspace model whenever the minimal distance $\arg\min_i ||\mathbf{x} - \mathbf{c}_i||$ of an achieved target $\mathbf{x}$ to the already explored targets (represented by the hyper-spheres $\mathbf{c}_i$) is larger than the hyper-sphere radius γ. The number of prototypes in the workspace model at episode e is denoted by $N(e)$.

The workspace model W serves two purposes: Firstly, it represents the explored workspace, which is useful during exploitation of the inverse model to check whether a reasonable performance can be expected for a desired target $\mathbf{x}^\star$. Secondly, it provides seeds for the local exploration mechanism, i.e. for sampling new targets by (1). Selection of exploration seeds $\mathbf{x}^{\text{seed}}$ can be implemented in various ways, e.g., preferring seeds in regions of the task space where errors are large [competence is low (Baranes and Oudeyer 2013)] or where sampling is expected to maximize skill improvement (active learning) similar to the work by da Silva et al. (2014b). In this paper, exploration seeds $\mathbf{x}^{\text{seed}}$ are drawn from the current set of prototypes $\{\mathbf{c}_i\}_{i=1,\ldots,N(e)}$ in the workspace model. The idea is to more frequently explore areas of the workspace that have been added to the task space model more recently. For this purpose, each prototype $\mathbf{c}_i$ is equipped with an age counter a_i that is initially set to 1 and incremented each time the prototype $\mathbf{c}_i$ served as exploration seed in (1). A new seed $\mathbf{x}^{\text{seed}} := \mathbf{c}_n$ is selected in each episode according to the probability

$$P_n = \frac{1/a_n}{\sum_{i=1}^{N(e)} 1/a_i}, \tag{7}$$

which prefers a uniform age distribution of the prototypes in the long run.

3.3 Summary of the skill babbling algorithm

The skill babbling algorithm is compactly presented in Algorithm 1. Note that parts of Algorithm 1 can be replaced by different implementations, e.g., the inverse model learner, the criterion for selecting exploration seeds, and the form of the distributions in (1) and (2). In this study, a restriction to simple, prototypical methods is deliberately chosen to exemplarily demonstrate the overall framework. It is noteworthy that this paper considers only tasks which can be expressed as inverse modeling problem. That is, a forward model needs to capture the relation between actions and effects.

4 Exploration of inverse kinematics

In this section, the autonomous exploration of direct inverse kinematics is demonstrated for planar arms with a varying number of degrees of freedom. The results show the validity of the concept and its scalability to high-dimensional action spaces. Moreover, the benefit of selecting a learner with non-local basis functions for rapid bootstrapping of inverse models is shown.

4.1 Planar arm with two degrees of freedom

First, autonomous exploration of inverse kinematics is exemplarily shown for a planar arm with two degrees of freedom. For illustration purposes and to show that the exploration algorithm copes with redundant solutions to the inverse kinematics problem, the targeted inverse model controls only the height x_2 of the end effector (see right column of Fig. 3). That is, the inverse model $\mathbf{y} = g(x_2)$ has a scalar input x_2 and a two-dimensional joint angle output $\mathbf{y} = (q_1, q_2)^T$. The setup is inspired by the work in Rolf et al. (2010) and allows for a complete visualization of the learning process.

Algorithm 1 Skill Babbling

```
Require: initial configuration $(\mathbf{x}^{\text{ini}}, \mathbf{y}^{\text{ini}})$
1: initialize task space model $W$ with $\mathbf{x}^{\text{ini}}$
2: initialize inverse model $g$ with $(\mathbf{x}^{\text{ini}}, \mathbf{y}^{\text{ini}})$
3: repeat
4:    Exploration:
5:      select exploration seed $\mathbf{x}^{\text{seed}} = \mathbf{c}_n$ according to (7)
6:      draw targets $\mathbf{x}_i^\star = \mathcal{N}(\mathbf{x}^{\text{seed}}, \sigma_{\text{tsk}}) \quad i = 1, \ldots, R$
7:      compute inverse estimates $\tilde{\mathbf{y}}_i = g(\mathbf{x}_i^\star) \quad i = 1, \ldots, R$
8:      add exploratory noise: $\mathbf{y}_i = \tilde{\mathbf{y}}_i + \mathcal{N}(0, \sigma_{\text{act}}) \quad \forall i$
9:      observe outcomes $\mathbf{x}_i = f(\mathbf{y}_i) \quad \forall i$
10:   Adaptation:
11:     compute weights $w_i \; \forall i$ according to (5)
12:     update inverse model $g(\mathbf{x}, \theta)$ with $\mathbf{x}_i, \mathbf{y}_i, w_i \quad \forall i$
13:     update workspace model $W$:
14:       • increment age counter $a_n$
15:       • add new prototype $\mathbf{c}^{\text{new}} = \mathbf{x}_l$ if $d_l > \gamma$ with
16:           $l = \arg\min_{i=1,\ldots,N(e)} d_i$ and $d_i = ||\mathbf{x}_i - \mathbf{x}_{\text{seed}}^\star||$
17: until task space model does not grow further
18: return inverse model $g$ and workspace model $W$
```

4.1.1 Experiment setup

Skill babbling is conducted with $R = 100$ roll-outs. With the choice of a simple workspace model and exploration strategy described in Sect. 3.2.2, the exploration noise variance in task space σ_{tsk} and the hyper-sphere radius γ in the workspace model can be set relative to each other. We set $\sigma_{\text{tsk}} = 0.75\gamma$ in (1) throughout this paper. The hyper-sphere radius γ is set to 0.2 in this example. An exploratory noise variance of 15 degrees is used in action space, i.e. $\sigma_{\text{act}} \approx 0.26$. For the inverse model $g(\mathbf{x}, \theta)$, a variant of the Extreme Learning Machine is applied with $H = 250$ hidden neurons, regularization parameter $\varepsilon = 0.001$, and forgetting factor $\lambda = 1 - 10^{-6}$ for the sequential learning rule. See Appendix for more details of the learning algorithm.

4.1.2 Results

Figure 3 shows snapshots of the autonomous exploration process, where the joint angle space together with the height of the end effector x_2 coded in gray-scale is shown in the left column. Arm configurations for the centers c_i of the workspace model are shown in the right column of Fig. 3. The initial condition $\mathbf{y}^{\text{ini}}$ is depicted by a magenta star and the centers c_i of the workspace models by circles in Fig. 3 (left). In each episode, the estimate of the inverse model (visualized by the solid lines in the left column of Fig. 3) improves step by step and is able to control the height x_2 of the end effector accurately with an average reaching error of 1.84 cm after 17 episodes. Note that the inverse model selects a solution to the inverse modeling problem which is in each point approximately perpendicular to the thin lines in Fig. 3 (left). This means that the inverse model follows the gradient of the forward function with respect to the actions, which is the most efficient solution in terms of the action-effect ratio. This efficient solution is enforced by the weighting scheme in (4), which prefers maximal changes in task space with minimal effort in action space. Note further that in the more complex examples presented later on it is assumed that the outcome of an action can only be observed either on the real plant or in a simulator. Numerical computation of the gradient to solve the inverse problem iteratively is extremely costly for these tasks. The direct inverse model $g(\mathbf{x})$ established after the learning procedure does immediately provide appropriate actions which achieve the desired outcome without repetitive estimation of the gradient.

The selected redundancy solution depends on the initial condition $(\mathbf{x}^{\text{ini}}, \mathbf{y}^{\text{ini}})$. The initial condition in Fig. 3 results in an elbow-up solution to the inverse kinematics (see Fig. 3 (right)). Figure 4 shows the bootstrapped inverse model for another initial condition which corresponds to an elbow-down solution. The initial condition in Fig. 4 resides close to a singularity of the forward function. Skill babbling neverthe-

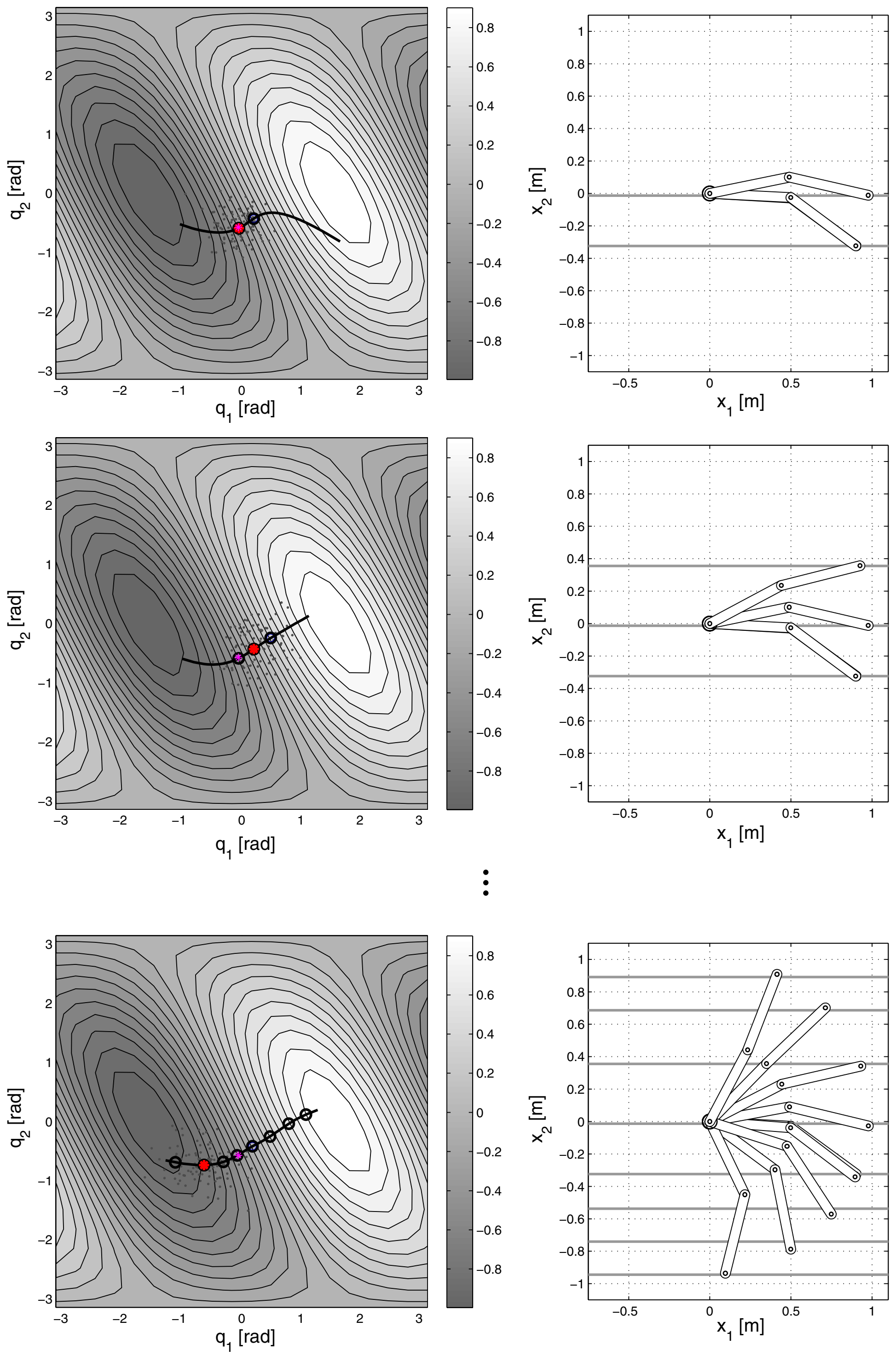


Fig. 3 Snapshots of the skill babbling process in episode 2 (*top row*), 3 (*middle row*) and 17 (*bottom row*) for learning the inverse kinematics of a planar arm with two degrees of freedom. The inverse model $\mathbf{y} = g(x_2)$ shall control the height of the end effector x_2. In the left column, the two-dimensional action space of joint angles $\mathbf{y} = (q_1, q_2)^T$ is shown. The outcome of the actions, i.e., height of the end effector x_2, is visualized in the left column by the gray-scale coding. *Thin lines* show actions with equal end effector heights. The *thick line* depicts the current inverse model $g(x)$, where the *circles* $g(c_i)$ are the centers c_i of the workspace model mapped to action space by the inverse model. *Small gray dots* show the current roll-outs around the exploration seed (*bold red star*). The *magenta star* indicates the initial demonstration $\mathbf{y}^{\text{ini}}$ in action space. In the *right column*, exemplary arm postures $g(c_i)$ obtained from the current inverse model are shown for the centers c_i of the workspace model (*gray, horizontal lines*) (Color figure online)

less succeeds in the discovery of an accurate inverse model within 20 episodes. In general, the initialization of skill babbling close to singularities is not problematic but may result in higher variance of the learned inverse model over several skill babbling trials.

4.2 Skill babbling with many degrees of freedom

In this section, the inverse kinematics scenario is scaled to planar arms with $N > 2$ degrees of freedom and control of the two-dimensional end effector position (see Fig. 5 for an

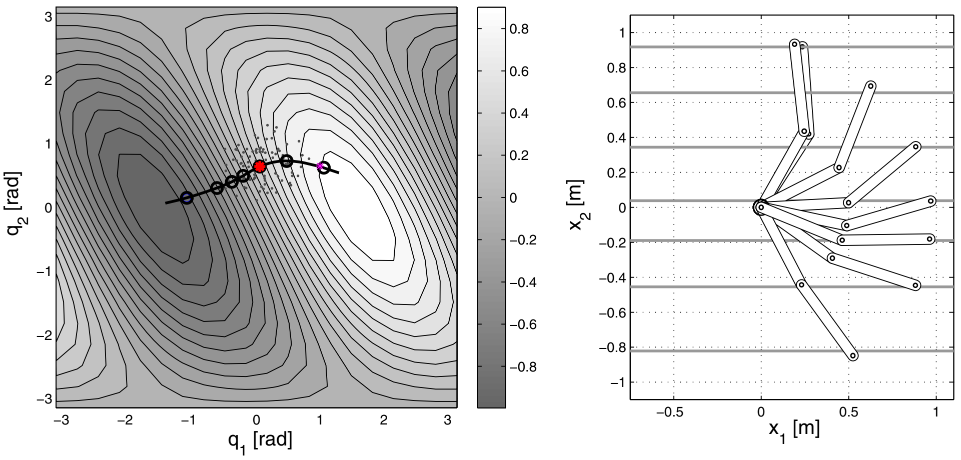


Fig. 4 Snapshot of the skill babbling process in episode 20 for learning the inverse kinematics of a planar arm with two degrees of freedom. The visualization is identical to Fig. 3, except for the initial condition $\mathbf{y}^{\text{ini}}$ (*magenta star* in the *left panel*). While the initial condition in Fig. 3 corresponds to an elbow-up solution, the initial condition in this skill babbling run is closer to a singularity (*bright area* in the *left panel*) and results in the selection of an elbow-down redundancy resolution (see *right panel*)

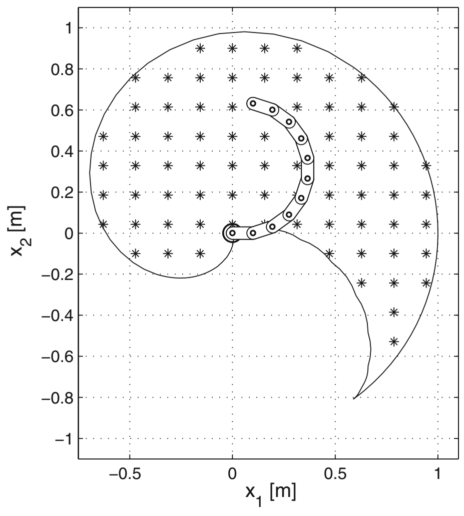


Fig. 5 Reaching errors (*left*) according to (8) (logarithmic scale of episodes). Workspace coverage (*right*) for the locally linear model (LLM) and extreme learning machine (ELM). *Solid lines* show median reaching errors and workspace coverage, respectively, and the *shaded areas* show the 25th and 75th percentiles over 10 independent skill babbling trials

exemplary arm with 10 DOFs). The considered arms have N links of length $1/N$ m each. The joint range is $[-0.3\pi, 0.3\pi]$ for the first joint and $[0, \frac{2\pi}{N-1}]$ for all other joints, which prohibits downward bending, but permits upward bending exactly to the extent that the robot can touch its own base.

4.2.1 Performance evaluation

To access the performance of the bootstrapped inverse kinematics, the accuracy of the learned inverse models is measured in task space. For a target $\mathbf{x}$ in task space, the reaching error is computed according to

$$e(\mathbf{x}) = ||\mathbf{x} - f(g(\mathbf{x}))||. \tag{8}$$

The reaching error (8) is computed for a representative set of targets $\{\mathbf{x}\}$ (the 76 targets shown by stars in Fig. 5) in the workspace after each episode, i.e. one cycle through lines 4–16 of Algorithm 1.

In addition to the reaching accuracy, the ability of the skill babbling algorithm to explore the entire workspace is evaluated. For this purpose, the area of the explored workspace is estimated numerically using alpha shapes (Edelsbrunner and Mücke 1994; Lundgren 2010). The explored workspace is estimated by computing the area of the alpha shape that is covered by all test points used in (8) for which the inverse model achieves a reaching error below 10 cm. Workspace coverage C is measured by the ratio between the area of the explored workspace and the area of the actual workspace. Hence, $C = 1$ indicates complete workspace coverage.

4.2.2 Experiment setup

Ten independent skill babbling trials are conducted with $R = 10$ roll-outs per episode. The initial condition $\mathbf{y}^{\text{ini}}$ is set to the center of joint space according to the joint angle limits, i.e. $\mathbf{y}^{\text{ini}} = (0, \frac{\pi}{N-1}, \ldots, \frac{\pi}{N-1})^T$ (see Fig. 5). The hyper-sphere radius γ for the workspace model is set to 0.1, which corresponds to hyper-disks in task space that are 20 cm in diameter. To account for the changing number of DOFs in the following experiments, the exploratory noise level σ_{act} in action space is selected depending on the number of joints as described in Rolf et al. (2011).

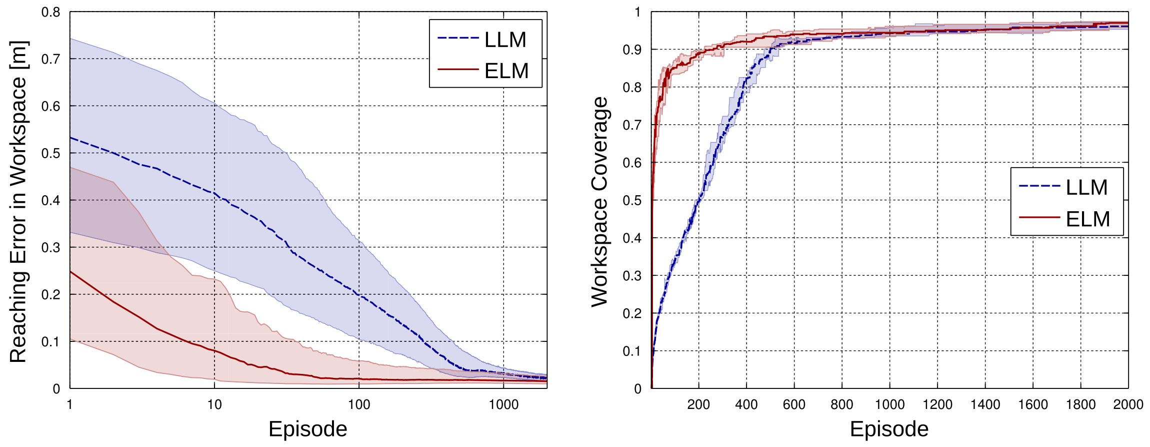


Fig. 6 Planar arm with ten degrees of freedom. The *arm* is shown in its home posture used to initialize learning. The *black solid line* encompasses the workspace for the joint angle ranges. The *black stars* depict the selected end effector positions used for evaluation of the reaching accuracy

For skill babbling, two different learners are used. Details of both learners are given in the Appendix. The first learner is a Locally Linear Model (LLM) that has been used in many goal babbling studies, e.g., Rolf et al. (2011) and Rolf and Steil (2014). The parameters of the LLM are set as follows: The radius d is set equal to the hyper-sphere radius γ of the workspace model, i.e. $d = \gamma = 0.1$. For parameter adaptation of the LLM by gradient descend, the learning rate $\eta = 0.25$ is used. The second learner is the ELM variant with identical parameters used in Sect. 4.1.1.

4.2.3 Non-local learner facilitates exploration

We first elaborate on the results of skill babbling for a $N = 10$ DOF arm. Figure 6 (left) shows the median of reaching errors (8) over ten independent skill babbling trials as function of the episode together with the 25th and 75th percentiles. The results show that the reaching errors decrease quickly in the beginning of skill babbling [logarithmically scaled abscissa in Fig. 6 (left)] and converges to a mean error of approximately 2 cm. While both learners achieve a similar final error niveau, skill babbling is much more rapid with the ELM learner. Note that the parameters of the LLM, such as the learning rate, have been tuned manually to achieve fast learning. However, the ELMs with their non-local feature encoding by the sigmoidal activation functions in the hidden layer seem to be beneficial for the autonomous exploration process. This is because each learning step results not only in local adaptation of the inverse model, but informs the inverse model on a spatially more extended scale about the structure of the inverse problem. This amplifies the positive feedback effect in the autonomous exploration process, which has already been described for online goal babbling by Rolf et al. (2011).

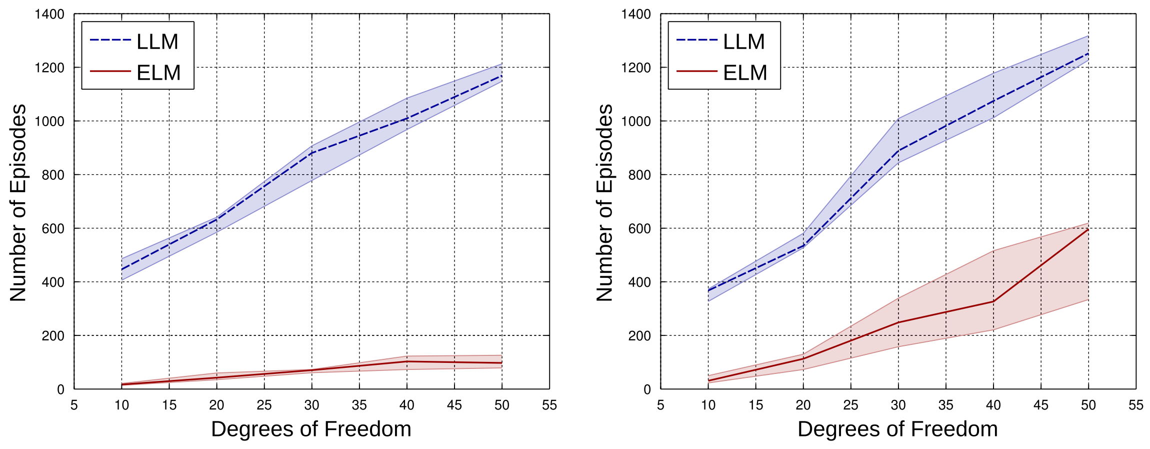


Fig. 7 Scaling of skill babbling to high-dimensional action spaces. Number of skill babbling episodes required to achieve median reaching errors below 5 cm (*left*) and workspace coverage above 80% (*right*) as function of the degrees of freedom N. Solid lines show median values and *shaded areas* show the 25th and 75th percentiles over 10 independent skill babbling trials

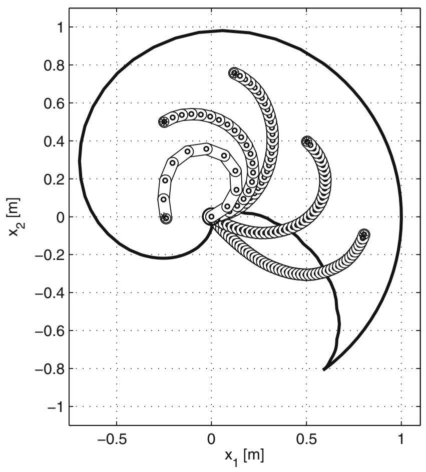


Fig. 8 Exemplary postures obtained from the learned inverse models (ELMs) for planar arms with 10, 20, 30, 40 and 50 degrees of freedom. Targets are shown by *black stars*

This positive effect of a learner with non-local basis functions is also confirmed by the workspace coverage plotted in Fig. 6 (right). This figure shows a much quicker workspace coverage by the ELM learner. Overall, the skill babbling algorithm achieves almost complete workspace coverage independent of the learner.

4.3 Linear scaling of learning speed

We finally investigate the scaling of skill babbling to high-dimensional action spaces. For this purpose, skill babbling is conducted for planar arms with $N \in \{10, 20, \ldots, 50\}$ degrees of freedom. For each experiment, the number of episodes required to achieve a desired performance level is measured. In particular, the number of episodes required to achieve a median error below 5 cm in the entire workspace and the number of episodes until 80% of the actual workspace is covered by the workspace model W are measured. The results of ten independent skill babbling trials for both measures are depicted in Fig. 7 and show that skill babbling scales approximately linearly with the number of DOFs for learning the inverse kinematics of the planar arms. That is, with increasing number of degrees of freedom, a linear increase of episodes until skill babbling is finished can be expected in this example. The slope of the increase is considerably higher for the LLM learner with respect to the number of episodes that are needed to reach the desired error niveau. This indicates again the positive effect of a learner with non-local basis functions by more quickly fitting the overall inverse model. While the variance of required episodes increases for both learners slightly with the number of DOFs, skill babbling robustly covers the workspace with an accurate inverse model in all cases. Exemplary arm postures obtained from the inverse models learned through skill babbling for 10, 20, 30, 40 and 50 degrees of freedom are shown in Fig. 8.

5 Exploration of parameterized motion primitives

The previous section demonstrated the effective and scalable bootstrapping of inverse kinematics with skill babbling. This section shows the ability of skill babbling to cope with more indirect and temporally extended relations between targets, actions, and outcomes. In particular, it is shown how the task-dependent parametrization of motion primitives can be learned autonomously. First, a state-of-the-art representation for motion primitives is introduced. Then, the learning of inverse models for the parametrization of handwriting motions by skill babbling is demonstrated. Finally, the scalability of skill babbling is shown for constrained reaching motions in a random bin picking scenario.

5.1 Motion primitive representation

In the last decade, a number of approaches that encode motion primitives in terms of dynamical systems have been proposed (e.g., see Lemme et al. (2015) for a collection of methods). In this paper, Dynamic Movement Primitives (DMPs, see Ijspeert et al. (2013) for a review) are used to represent motions within a second-order dynamical system formulation. Motions are generated by DMPs according to the perturbed spring-damper system

$$\tau \ddot{\mathbf{u}} = \kappa(\mathbf{g} - \mathbf{u}) - \delta \dot{\mathbf{u}} + p(c), \tag{9}$$

where τ is a time constant, $\mathbf{g}$ is the terminal state of the motion, κ and δ are spring and damping parameters, respectively. The perturbation $p(\cdot)$ allows for modulation of the motion shape and velocity profile. The perturbation $p(c)$ is a function of a phase variable c, which encodes the temporal progress of the motion. In the preparation of a discrete motion, c is typically set to 1 and then decays exponentially. In contrast to the typical exponential decay of c for discrete motions in the literature, we use a constantly decreasing system

$$\dot{c} = \begin{cases} 0, & \text{if } c \leq 0 \\ -\frac{\alpha}{\tau}, & \text{otherwise} \end{cases} \tag{10}$$

similar to Kulvicius et al. (2012). Eq. (10) results in very robust learning of $p(\cdot)$ and can be intuitively interpreted as the expected progress of the motion (Reinhart and Steil 2014, 2015). For properly chosen spring and damping constants, stability of the generated motions is guaranteed by constructing the perturbation such that $p(0) = \mathbf{0}$ at the end of the motion.

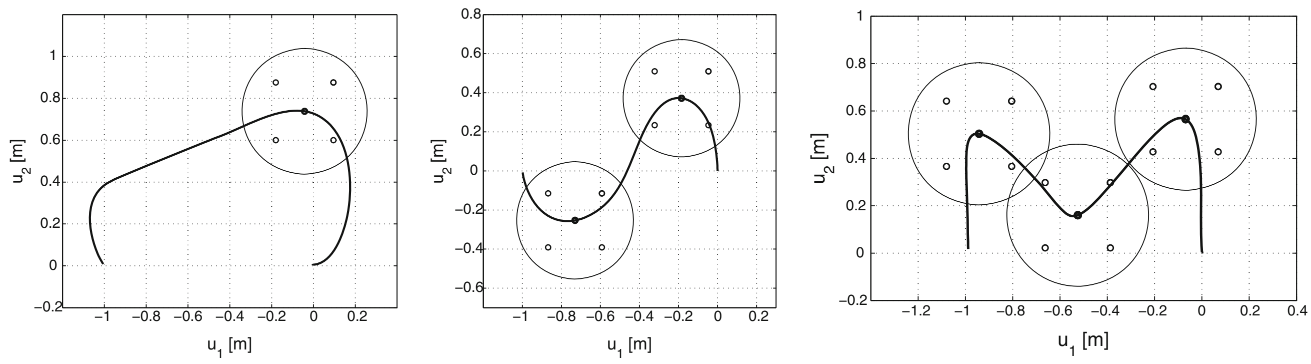


Fig. 9 Example motions (*solid black lines*) used as initial demonstrations for skill babbling: Arc-shape (*left*), S-shape (*center*), and M-shape (*right*). Via points (*circles*) have been computed by a minimum velocity criterion on the demonstrations. The other *circles* are novel target via points used for evaluation of the parameterized skills. Note that all combinations of test via points are used for the motions with more than one via point. The *thin circles* are the regions $\mathcal{R}$ considered in the task space constraint (15) for this example

In the following, the particular form of $p(\cdot)$ and its supervised training from a demonstration is described.

5.1.1 Encoding of motion shapes in DMPs

Typically, the perturbation $p(c)$ is implemented by a linear model with Gaussian basis functions $\varphi_i(c) = exp(-\frac{1}{2\rho_i^2}||c - b_i||^2)$:

$$p(c) = \mathbf{W}^{\text{DMP}} \boldsymbol{\varphi}(c). \tag{11}$$

The DMP parameters $\mathbf{W}^{\text{DMP}} \in \mathbb{R}^{D\times G}$ encode the motion shape, where $G = \dim(\boldsymbol{\varphi})$ is the number of basis functions. Because of the constantly decreasing phase variable c given by (10), it is for most motions suffcient to distribute the basis function centers b_i equally spaced on the interval [0, 1] and use identical basis function widths $\rho_i \equiv \rho$.

5.1.2 Supervised training of DMPs

To provide seeds for the autonomous skill exploration later on, initial motions are trained from demonstrations in a supervised manner. Given an initial demonstration of a motion $\mathbf{u}(k), k = 1, \dots, K$ with K steps, targets for adopting the DMP parameters $\mathbf{W}^{\text{DMP}}$ of the linear model (11) by supervised learning are derived by numerically calculating the first and second derivatives of $\mathbf{u}(k)$ with respect to time. Then, the targets $p^*(c(k))$ for each time step k are obtained by rearranging (9):

$$p^*(c(k)) = -\kappa(\mathbf{g} - \mathbf{u}(k)) + \delta\dot{\mathbf{u}}(k) + \tau\ddot{\mathbf{u}}(k). \tag{12}$$

Learning can be accomplished by linear regression

$$\mathbf{W}^{\text{DMP}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{P}, \tag{13}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{K\times G}$ are the collected basis function activations for all time steps k and $\mathbf{P}$ the respective targets for $p(\cdot)$. The parameter α in (10) can be set to $1/(K+1)$.

5.1.3 Integration of DMPs in skill babbling

The DMP parameter matrix $\mathbf{W}^{\text{DMP}}$ modulates the shape of the motion generated by (9). For autonomous exploration of a parameterized skill, an inverse model $\boldsymbol{\omega} = g(\mathbf{x})$ is bootstrapped, where the outputs $\boldsymbol{\omega} \in \mathbb{R}^{G\cdot D}$ are a vectorized format of the DMP parameter matrix $\mathbf{W}^{\text{DMP}}$. Additional DMP properties, e.g., time constants, can be integrated into the vectorized representation. In Algorithm 1 (line 9), actions $\mathbf{y} \equiv \boldsymbol{\omega}$ (policy parameters) are unrolled in time by executing a DMP with the parameters $\boldsymbol{\omega}$ before the forward model is evaluated (cf. Fig. 2).

5.2 Learning parameterized via point motions

In a first example, the autonomous exploration of parameterized handwriting motions through via points is demonstrated. Three exemplary motions from the data set recorded by Khansari-Zadeh (2012) at the LASA institute at EPFL are preprocessed to have start and terminal positions at $\mathbf{u}(1) = (-1, 0)^T$ and $\mathbf{u}(K = 150) = (0, 0)^T$, respectively (see Fig. 9). In addition, the via points $\mathbf{v}_i$ of these demonstrations are computed based on a minimum velocity criterion. The first motion has a single via point $\mathbf{v}$ (Fig. 9 (left)), the second S-shaped motion has two via points $\mathbf{v}_1$ and $\mathbf{v}_2$ (Fig. 9 (center)). The third motion is M-shaped and has three via points $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ (Fig. 9 (right)).

The goal is to learn an inverse model which allows for a parametrization of the via points. That is, given a desired via point $\mathbf{v}^\star$, the inverse model $g(\mathbf{v}^\star)$ shall return actions (DMP parameters $\boldsymbol{\omega}$), such that the unrolled motion passes through $\mathbf{v}^\star$. Thus, the forward model computes the actual via points $\mathbf{v}_i = \mathbf{u}(k_i^v)$ by returning the position along the generated motion $\mathbf{u}(k)$ obtained from $\boldsymbol{\omega} = g(\mathbf{v}^\star)$ at the desired traversal times k_i^v. To accommodate multiple via points in this notation, we define the vector of concatenated via points $\boldsymbol{\nu} = (\mathbf{v}_1^T, \ldots, \mathbf{v}_V^T)^T \in \mathbb{R}^{V \cdot dim(\mathbf{u})}$, where V is the number of parameterized via points along the motion. The forward model then computes $\boldsymbol{\nu} = (\mathbf{u}(k_1^v)^T, \ldots, \mathbf{u}(k_V^v)^T)^T$ accordingly.

5.2.1 Skill babbling of via point motions

For each motion, a DMP with $\tau = 1, \kappa = 0.1, \delta = 2\sqrt{\kappa}$ and $G = 10$ is trained according to Sect. 5.1.2. The vectorized DMP parameters $\boldsymbol{\omega}$ together with the vector of concatenated via points $\boldsymbol{\nu}$ serve as initial condition for skill babbling. That is, $\mathbf{x}^{\text{ini}} = \boldsymbol{\nu}$ and $\mathbf{y}^{\text{ini}} = \boldsymbol{\omega}$.

The basic weighting scheme introduced in Sect. 3.2.1 is sufficient to successfully explore the considered parameterized via point motion skills. However, in order to foster smoother motions and to demonstrate the extension of the weighting scheme, a minimum jerk weighting criterion is accommodated into (5). For this purpose, the absolute jerks $||\dddot{\mathbf{u}}(k)||$ of unrolled trajectories $\mathbf{u}(k)$ are integrated. With a slight abuse of notation, we compute

$$J(\mathbf{u}(k)) = \sum_{k=1}^{K} ||\dddot{\mathbf{u}}(k)||.$$

The weights w_i^{minjerk} for all roll-outs i in an episode are then normalized by the soft-max operation

$$w_i^{\text{minjerk}} = \frac{exp(-\alpha J(\mathbf{u}_i(k))}{\sum_{j=1}^{R} exp(-\alpha J(\mathbf{u}_j(k)))}, \tag{14}$$

where the factor α is set to 100 in order to achieve a pronounced decision for smooth trajectories.

Note further that the workspace is unbounded for this task. In order to bound the exploration in time (give skill babbling a chance to finish exploration) and to bound the workspace spatially, an additional weighting criterion is factored in (5). It implements a spatial task constraint and computes binary weights

$$w_i^{\text{tskcnst}} = \begin{cases} 1 & \text{if } f(g(\mathbf{x})) \in \mathcal{R} \\ 0 & \text{otherwise} \end{cases}, \tag{15}$$

where $\mathcal{R}$ defines a region in task space. This spatial task constraint effectively turns off learning outside a predefined region $\mathcal{R}$ in task space. For the via point tasks, circular regions $\mathcal{R}$ with radius 0.3 are defined around each via point (see Fig. 9).

Ten independent skill babbling trials are conducted with $R = 20$, $\gamma = 0.1$, and $\sigma_{\text{act}} = 0.1 \cdot \text{std}(\mathbf{y}^{\text{ini}})$ for each motion from Fig. 9. For inverse model training, an ELM with the same parameters as in Sect. 4.1.1 is selected. To account for the varying input dimensions of the inverse model depending on the number of via points V, the input weights $\mathbf{W}^{\text{inp}}$ of the ELM are initialized in in range $[-2/V, 2/V]$ (see Appendix).

5.2.2 Results

Figure 11 shows the via point error statistics (median and 25th and 75th percentiles) as function of the skill babbling episode. The via point errors

$$e_i = ||\mathbf{v}_i - f(g(\mathbf{v}_i))|| \quad \forall i = 1, \ldots, V \tag{16}$$

decrease as skill babbling proceeds and reaches mean errors of 1.8, 0.8, and 1.7 cm for the arc-, S-, and M-shaped trajectories, respectively, which corresponds to accurate via point traversal. Exemplary trajectories for selected via points are shown in Fig. 10. The trajectories are smooth and the shape of the initial demonstration is preserved. Note that preservation of the motion shape is actually equivalent to preserving a particular redundancy resolution over the workspace in this scenario.

Note further that for each example trajectory, skill babbling learns an inverse model which encodes an entire set of parameterized motions. In contrast to common policy improvement methods with DMPs (Theodorou et al. 2010; Stulp and Sigaud 2013; Kormushev et al. 2010), where a dedicated optimization process is conducted for each via point, motions can be immediately retrieved from the inverse model for arbitrary via points in the workspace. Additionally, this example shows that skill babbling scales also to higher-dimensional task spaces: For the M-shaped trajectory, there are three two-dimensional via points. Hence, the inverse model has six inputs. Skill babbling successfully learns a parameterized motor skill also in this case of a rather complex and high-dimensional motion parametrization.

5.3 Learning constrained robot motions

In a final demonstration, skill babbling is applied to a simplified random bin picking scenario, i.e. vision-guided picking of randomly distributed objects from a container. The scenario is as follows: A robotic manipulator with 7 DOF operates in a production cell with fixed bin position and additional workspace constraints (table and ceiling, see Fig. 12). For simplicity, the target object position $\mathbf{x}$ is assumed to be given and we neglect object orientation. The manipulator starts from the same initial configuration and has to reach the object at $\mathbf{x}$ without colliding with the environment. Hence,

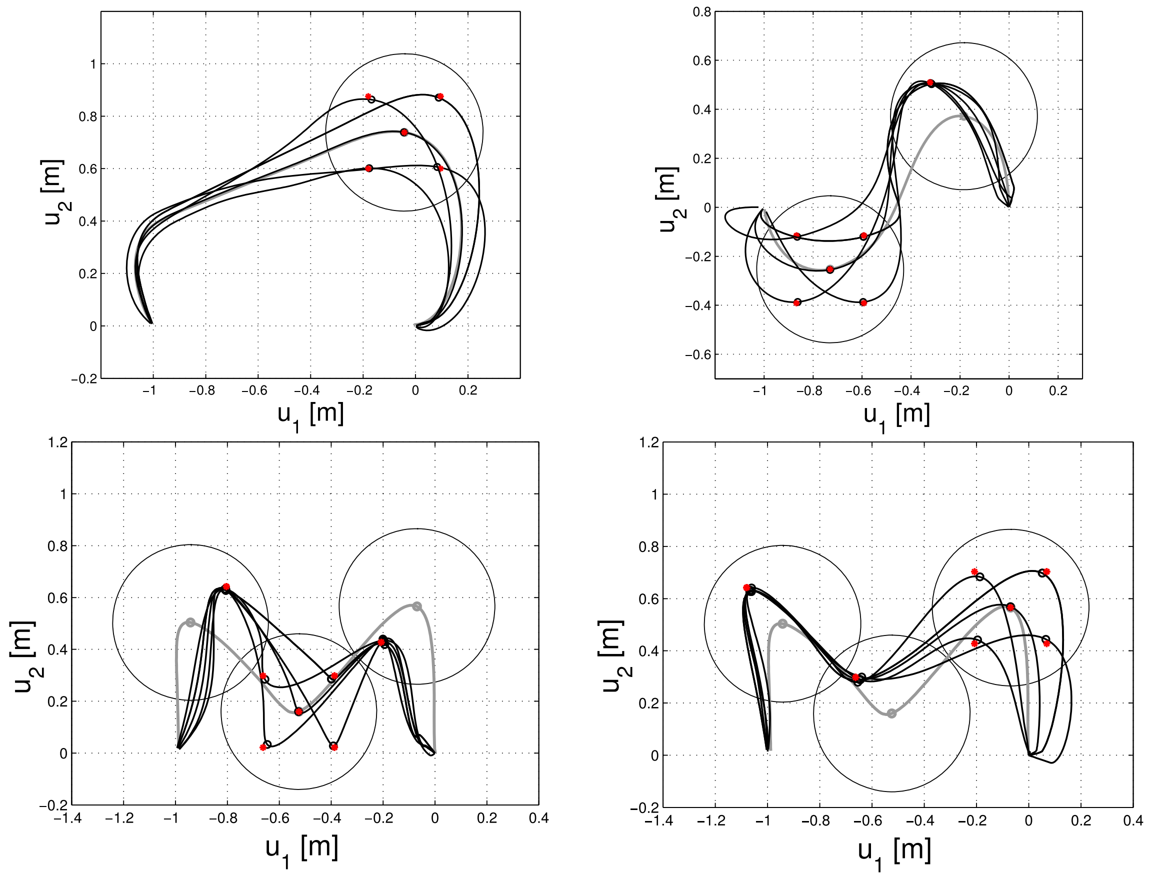


Fig. 10 Exemplary motions obtained from the autonomously explored motor skills. The initial demonstrations are shown by *solid gray lines*. Trajectories generated from the DMP with the estimated DMP parameters $\boldsymbol{\omega} = g(\boldsymbol{\nu})$ for novel via points $\boldsymbol{\nu}$ are shown by *solid black lines*. Target via points are depicted by *red stars* and actual via points by *black circles*. For the M-shaped trajectory, motions for different via point combinations are shown in the *bottom* panels (Color figure online)

the object position $\mathbf{x} \in \mathbb{R}^3$ parameterizes the random bin picking skill.

5.3.1 Initial demonstration of bin picking

For skill babbling, we teach-in a single demonstration with a simple Cartesian interface. Motions $\mathbf{u}(k)$ are represented in joint angle space in order to allow for immediate motion execution given a new target $\mathbf{x}$. The initial demonstration is encoded in a DMP, which operates in joint angle space, i.e. $\mathbf{u}(k) \equiv \mathbf{q}(k) \in \mathbb{R}^7$ for the KUKA LWR. The DMP spring-damper and time constants are set identical to the previous section. The demonstration is collision-free and shown in Fig. 12.

5.3.2 Skill babbling for random bin picking

The demonstration serves as initial condition for skill babbling, which aims at finding an inverse model $\mathbf{y} = g(\mathbf{x})$ that maps target positions to parameters of the DMP. While

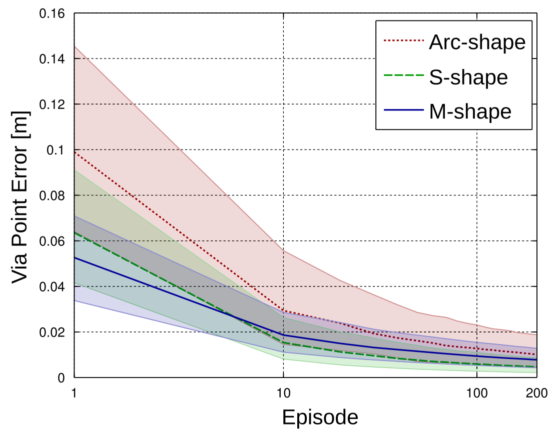


Fig. 11 Median of the via point traversal error (16) over skill babbling episodes (logarithmic scale) for motions with one (Arc-shape), two (S-shape) and three (M-shape) via points

the task space is three-dimensional, the space of policy parameters is much higher dimensional than in the previous examples. The DMP controls 7 joint angles with $G = 10$

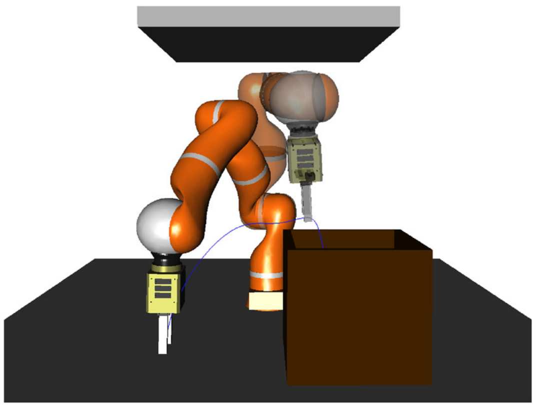


Fig. 12 Simplified random bin picking scenario with a KUKA LWR arm in a confined workspace. The initial demonstration is collision-free and depicted by the *blue* end effector trajectory. The size of the bin is 30 × 30 × 30 cm (Color figure online)

basis functions per joint. In addition, the terminal state of the DMP $\mathbf{g}$ in (9) is parameterized by the inverse model. In total, the parameterized skill $g(\mathbf{x})$ maps targets $\mathbf{x} \in \mathbb{R}^3$ to policy parameters $\mathbf{y} = (\boldsymbol{\omega}^T, \mathbf{g})^T \in \mathbb{R}^{7\cdot(G+1)}$ in a 77-dimensional space. Note that the terminal state $\mathbf{g}$ can be also controlled by an inverse kinematics solver. However, this example shows that skill babbling also copes with the combined problem of motion shape and inverse kinematics learning.

Three independent skill babbling trials are conducted with $\gamma = 0.02$, $R = 10$ and a manually tuned action noise distribution

$$\mathcal{N}\left(\mathbf{0}, \mathrm{diag}(0.1 \cdot \mathrm{std}(\boldsymbol{\omega}) \cdot \mathbf{1}_{70}, 0.05 \cdot \mathbf{1}_7)\right)$$

in (2), where $\mathbf{1}_D$ denotes a D-dimensional row vector. For training of the inverse model $g(\mathbf{x})$, ELMs with same parameters as in Sect. 4.1.1 are used.

In addition to the basic weighting scheme from Sect. 3.2.1, the following three criteria are injected into the autonomous exploration process:

- The constrained task space region criterion from Sect. 5.2.1 is applied, where the region $\mathcal{R}$ in (15) is restricted to the inner cuboid of the bin (see Fig. 12).
- Minimal path length in joint space is preferred for efficient motion. This is accomplished by computing

$$w_i^{\mathrm{len}} = \frac{exp\left(-\alpha L(\mathbf{u}_i(k)\right)}{\sum_{j=1}^{R} exp\left(-\alpha L(\mathbf{u}_j(k))\right)},$$

  similar to (14) but with $\alpha = 1$ and the path length measure $L(\mathbf{u}(k)) = \sum_{k=2}^{K} ||\mathbf{u}(k) - \mathbf{u}(k-1)||$.
- Finally, obstacle avoidance is integrated into the weighting scheme (5) by computing

$$w_i^{\mathrm{obs}} = \begin{cases} 1 & \text{if motion } \mathbf{u}_i(k) \text{ is collision-free} \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

  Obstacle collisions are evaluated based on the geometric model of the manipulator and the environment.

5.3.3 Results

The parameterized skill $g(\mathbf{x})$ is compared to the performance of the initial DMP together with an inverse kinematics solver to adopt the terminal joint angle configuration $\mathbf{g}$ in (9). While the reaching accuracy with the inverse kinematics solver is high, only 17% of the motions generated with the initial DMP to 64 target positions inside the bin are collision-free. It is noteworthy that these collisions occur during the motion, e.g., while passing the rim of the bin, and are not due to the redundancy resolution selected by the inverse kinematics

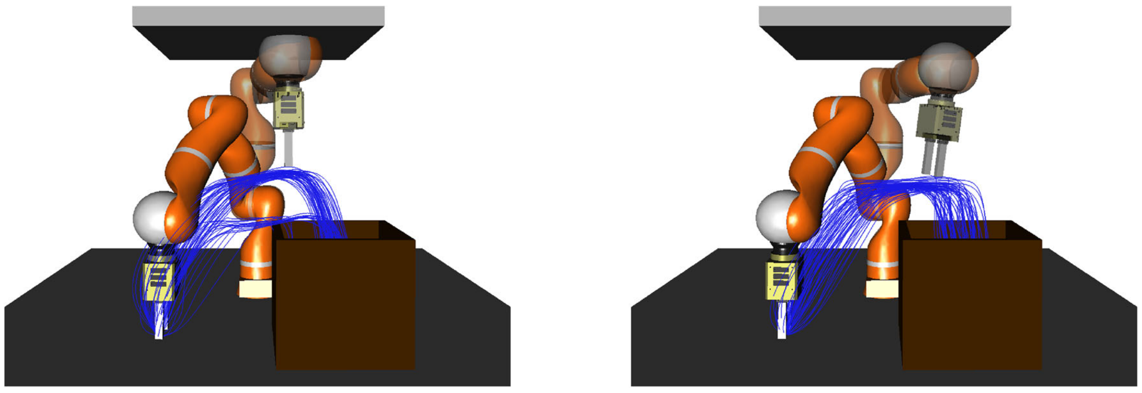
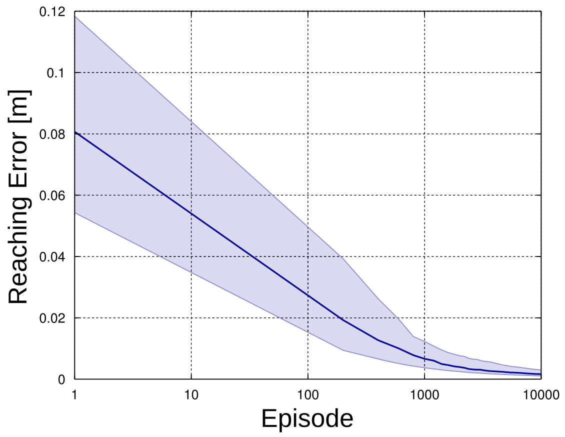


Fig. 13 Motion generation to 64 targets on a regular grid in the bin using the initial DMP together with an inverse kinematics solver (*left*) and using the parameterized skill learned by skill babbling (*right*). Usage of the initial DMP with inverse kinematics solving causes many collisions with the ceiling (see exemplary arm configuration in the left picture) and the bin. The parameterized skill learned by skill babbling generates collision-free motions to the targets in the bin (*right*)



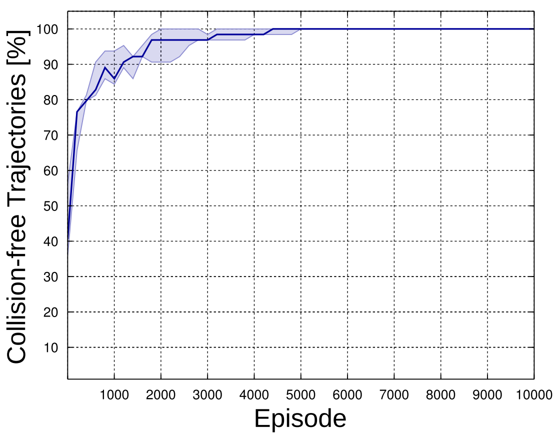


Fig. 14 Reaching errors (*left*) according to (8) for the final target positions in the bin (logarithmic scaling of episodes). Percentage of collision-free motions to the targets in the bin (*right*). *Solid lines* show median values and the *shaded areas* show the 25th and 75th percentiles in both graphs over 3 independent skill babbling trials

solver for the terminal arm configuration inside the bin (see Fig. 13 (left)).

Skill babbling successfully adopts the DMP parameters $\boldsymbol{\omega}$ encoding the motion shape in joint space depending on the target $\mathbf{x}$ in the bin such that no collisions occur for all 64 targets in the bin after approximately 5000 episodes (see Fig. 14 (right)). The reaching accuracy is measured in task space according to (8) for the terminal configuration of the arm contained in the seven last components of $\mathbf{y}$. The reaching error is very low and shows the high accuracy that can be achieved with skill babbling (see Fig. 14 (left)). Figure 14 shows also that skill babbling first adapts the parameterized skill such that collisions are avoided and then fine-tunes the reaching accuracy.

Figure 13 (right) shows the end effector motions for the joint angles trajectories generated by the parameterized skill. Skill babbling learns a smooth modulation of joint angle motions to avoid collisions with the bin and the ceiling. It is noteworthy that skill babbling copes with the composed task of solving on the one hand the inverse kinematics for the terminal DMP position ($\mathbf{g} = \tilde{g}_1(\mathbf{x})$), and on the other hand finding a collision-free path through configuration space ($\boldsymbol{\omega} = \tilde{g}_2(\mathbf{x})$).

6 Conclusion

Skill babbling is a novel framework to autonomous exploration of motor skills. It roots in the learning of inverse models by goal babbling, i.e. the sample-efficient bootstrapping of inverse models by organizing exploration in task space. Its episodic scheduling allows for higher learning rates compared to goal babbling, because a normalization of weights for the regression can be computed per episode, and it separates concerns: Exploration mechanism, adaptation of the inverse model including the weighting scheme, and the acquisition of a workspace model constitute separate algorithmic components of the skill babbling framework. While in this paper each of these components was deliberately chosen to be of simple form, future studies may investigate and exploit properties of more advanced, e.g., probabilistic, representations within the skill babbling framework.

In contrast to dedicated policy optimization for single task parametrizations as typically considered in reinforcement learning, skill babbling learns complete inverse models to cope with the variety of a skill which makes re-optimization for novel task parametrizations dispensable. This paper shows for the first time how to consistently learn inverse models for parameterized motion primitives starting from a single demonstration with subsequent autonomous training sample generation based on a guided exploration mechanism. The results presented in this paper further underline the robustness and excellent scalability of skill babbling to high-dimensional skill parametrizations.

Future directions of research include adaptive noise distributions for exploration. In particular an adaptive covariance matrix scheme for exploration in action space will mitigate the need to chose exploratory noise levels manually.

Acknowledgements This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster Competition and managed by the Project Management Agency Karlsruhe (PTKA).

Appendix: Learning algorithms

For skill babbling, any learning algorithm is applicable which solves the weighted regression problem (6) online. In this paper, the two following learning algorithms are applied.

Locally linear model

The first learner is a Locally Linear Model (LLM, Ritter 1991) as it is implemented by Rolf et al. (2011). It comprises $l = 1, \ldots, L$ linear models

$$g_l(\mathbf{x}) = \mathbf{W}_l\mathbf{x} + \mathbf{b}_l.$$

The linear models $g_l(\mathbf{x})$ are combined according to

$$\mathbf{y}(\mathbf{x}) = \frac{1}{n(\mathbf{x})}\sum_{l=1}^{L} b\left(\frac{\mathbf{x}-\mathbf{p}_l}{d}\right) g_l\left(\frac{\mathbf{x}-\mathbf{p}_l}{d}\right)$$

with Gaussian responsibilities

$$b(\mathbf{x}) = exp(-||\mathbf{x}||^2)$$

and the normalization

$$n(\mathbf{x}) = \sum_{l=1}^{L} b\left(\frac{\mathbf{x}-\mathbf{p}_l}{d}\right),$$

where the radius d determines the area of responsibility for each linear model around prototypical centers $\mathbf{p}_l$.

For supervised training, appropriate centers $\mathbf{p}_l$ for the linear models have to be found and the parameters $\mathbf{W}_l$ and $\mathbf{b}_l$ of the linear models have to be learned. Following the implementation by Rolf et al. (2011), centers are incrementally added to the learner according to a vector quantization algorithm. For the initial training sample $(\mathbf{x}_1, \mathbf{y}_1)$, the first linear model with center $\mathbf{p}_1 = \mathbf{x}_1$, linear part $\mathbf{W}_1 = \mathbf{0}$, and bias $\mathbf{b}_1 = \mathbf{y}_1$ is created. For new training samples $(\mathbf{x}_n, \mathbf{y}_n)$ together with the weight w_n, the weighted square error $w_n||\mathbf{y}_n - \mathbf{y}(\mathbf{x}_n)||^2$ is minimized by online gradient descend with learning rate η.

Extreme learning machine with weighted recursive least squares learning

The second learner is a variant of an Extreme Learning Machine (ELM, Huang et al. 2004) with weighted recursive least squares learning. ELMs are feedforward neural networks with a single hidden layer. The output is computed according to

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^{\text{out}T}\boldsymbol{\sigma}(\mathbf{W}^{\text{inp}}\mathbf{x} + \mathbf{b}),$$

where $\sigma(a) = 1/(1 + exp(-a))$ is a sigmoid activation function applied component-wise to the synaptic summations $\mathbf{a} = \mathbf{W}^{\text{inp}}\mathbf{x} + \mathbf{b}$. The special property of ELMs is that learning is restricted to the read-out weights $\mathbf{W}^{\text{out}}$, which makes backpropagation of errors dispensable. Learning then boils down to a simple linear regression problem for $\mathbf{W}^{\text{out}}$. The input weights $\mathbf{W}^{\text{inp}} \in \mathbb{R}^{H\times dim(\mathbf{x})}$ and biases $\mathbf{b} \in \mathbb{R}^H$ are initialized randomly and remain fixed. Typically, the number of hidden neurons H is chosen large in comparison to the number of inputs. In this paper, the values of $\mathbf{W}^{\text{inp}}$ and $\mathbf{b}$ are drawn from uniform distributions in range $[-2, 2]$ and $[-1, 1]$ if not stated otherwise. Note that the idea of using feedforward neural networks with a random hidden layer has been proposed earlier, e.g., by Schmidt et al. (1992).

While the variant of ELMs proposed by Liang et al. (2006) does feature sequential learning, it does not incorporate a weighted error criterion and does not make use of a forgetting factor. In this paper, a weighted recursive least squares algorithm similar to Haykin (1991) is applied in order to update the read-out weights $\mathbf{W}^{\text{out}}$ sequentially. For the first training sample $(\mathbf{x}_1, \mathbf{y}_1)$, the read-out weights are initialized according to

$$\mathbf{W}_1^{\text{out}} = \mathbf{P}_1\mathbf{h}(\mathbf{x}_1)\mathbf{y}_1^T \quad \text{with} \quad \mathbf{P}_1 = (\mathbf{h}(\mathbf{x}_1)\mathbf{h}(\mathbf{x}_1)^T + \varepsilon\mathbb{1}_H)^{-1},$$

where

$$\mathbf{h}(\mathbf{x}) = \sigma(\mathbf{W}^{\text{inp}}\mathbf{x} + \mathbf{b}) \ \in \mathbb{R}^H$$

are the hidden neuron activations for input $\mathbf{x}$ and $\varepsilon > 0$ is a regularization parameter. For new training samples $(\mathbf{x}_n, \mathbf{y}_n)$ together with the weight w_n, the read-out weights are updated sequentially according to the weighted recursive least squares rule

$$\begin{aligned}
\boldsymbol{\kappa}_n &= \frac{1}{\lambda + \mathbf{h}(\mathbf{x}_n)^T\mathbf{P}_{n-1}\mathbf{h}(\mathbf{x}_n)}\mathbf{P}_{n-1}\mathbf{h}(\mathbf{x}_n)\\
\mathbf{W}_n^{\text{out}} &= \mathbf{W}_{n-1}^{\text{out}} + w_n\boldsymbol{\kappa}_n\left(\mathbf{y}_n - \mathbf{W}_{n-1}^{\text{out}\,T}\mathbf{h}(\mathbf{x}_n)\right)\\
\mathbf{P}_n &= \frac{1}{\lambda}\left(\mathbf{P}_{n-1} - \boldsymbol{\kappa}_n^T\mathbf{h}(\mathbf{x}_n)\mathbf{P}_{n-1}\right),
\end{aligned}$$

where $0 \ll \lambda \leq 1$ is a forgetting factor. That is, $\lambda = 1$ corresponds to no forgetting.

References

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Baranes, A., & Oudeyer, P. Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, *61*(1), 49–73.

Calinon, S., Alizadeh, T., & Caldwell, D. G. (2013). On improving the extrapolation capability of task-parameterized movement models. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 610–616).

Edelsbrunner, H., & Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, *13*(1), 43–72.

Haykin, S. (1991). *Adaptive filter theory*. New York: Prentice Hall.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004). Extreme learning machine: A new learning scheme of feedforward neural networks. *IEEE International Joint Conference on Neural Networks*, *2*, 985–990.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, *16*(3), 307–354.

Khansari-Zadeh, S. (2012). http://www.amarsi-project.eu/open-source

Kober, J., Wilhelm, A., Oztop, E., & Peters, J. (2012). Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, *33*, 361–379.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, *32*(11), 1238–1274.

Kormushev, P., Calinon, S., & Caldwell, D. (2010). Robot motor skill coordination with EM-based reinforcement learning. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 3232–3237).

Kulvicius, T., Ning, K., Tamosiunaite, M., & Worgötter, F. (2012). Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, *28*(1), 145–157.

Kupcsik, A., Deisenroth, M. P., Peters, J., & Neumann, G. (2013). Data-efficient generalization of robot skills with contextual policy search. In *AAAI conference on artificial intelligence* (pp. 1401–1407).

Lemme, A., Meirovitch, Y., Khansari-Zadeh, S. M., Flash, T., Billard, A., & Steil, J. J. (2015). Open-source benchmarking for learned reaching motion generation in robotics. *Paladyn Journal of Behavioral Robotics*, *6*(1), 30–41.

Liang, N. Y., Huang, G. B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, *17*(6), 1411–1423.

Lundgren, J. (2010). `alphavol.m`. http://au.mathworks.com/matlabcentral/fileexchange/28851-alpha-shapes

Matsubara, T., Hyon, S., & Morimoto, J. (2010). Learning stylistic dynamic movement primitives from multiple demonstrations. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1277–1283).

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *Intern Journal of Robotics Research*, *32*(3), 263–279.

Pontón, B., Farshidian, F., & Buchli, J. (2014). Learning compliant locomotion on a quadruped robot. In I. R. O. S. Workshop (Ed.), *Compliant manipulation: Challenges in learning and control*.

Reinhart, R., & Steil, J. (2014). Efficient policy search with a parameterized skill memory. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1400–1407).

Reinhart, R. F., & Steil, J. J. (2015). Efficient policy search in low-dimensional embedding spaces by generalizing motion primitives with a parameterized skill memory. *Autonomous Robots*, *38*(4), 331–348.

Ritter, H. (1991). Learning with the self-organizing map. In *Artificial neural networks* (pp. 357–364). New York: Elsevier.

Rolf, M., & Steil, J. (2014). Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(6), 1147–1160.

Rolf, M., Steil, J., & Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, *2*(3), 216–229.

Rolf, M., Steil, J., & Gienger, M. (2011). Online goal babbling for rapid bootstrapping of inverse models in high dimensions. *IEEE International Conference on Development and Learning*, *2*, 1–8.

Schmidt, W., Kraaijveld, M., & Duin, R. (1992). Feedforward neural networks with random weights. In *IAPR international conference on pattern recognition, conference B: Pattern recognition methodology and systems* (Vol. II, pp. 1–4).

da Silva B. C., Konidaris, G., & Barto, A. G. (2012). Learning parameterized skills. In *International conference on machine learning* (pp. 1679–1686).

da Silva B. C., Baldassarre, G., Konidaris, G., & Barto, A. (2014a) Learning parameterized motor skills on a humanoid robot. In *IEEE international conference on robotics and automation* (pp. 5239–5244).

da Silva, B. C., Konidaris, G., & Barto, A. (2014b) Active learning of parameterized skills. In *International conference on machine learning, JMLR workshop and conference proceedings* (pp. 1737–1745).

Stulp, F., & Sigaud, O. (2013). Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn Journal of Behavioral Robotics*, *4*(1), 49–61.

Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., & Sigaud, O. (2013). Learning compact parameterized skills with a single regression. In *IEEE-RAS international conference on humanoid robots* (pp. 417–422).

Theodorou, E., Buchli, J., & Schaal, S. (2010). A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, *11*, 3137–3181.

Ude, A., Riley, M., Nemec, B., Kos, A., Asfour, T., & Cheng, G. (2007). Synthesizing goal-directed actions from a library of example movements. In *IEEE-RAS international conference on humanoid robots* (pp. 115–121).

René Felix Reinhart received his diploma in computer science at Bielefeld University, Germany, in 2008. In 2011, F. Reinhart received a Ph.D. for his subsequent research at the Bielefeld Institute for Cognition and Robotics (CoR-Lab). F. Reinhart was a visiting researcher at the Advanced Robotic Controls Group at NASA JPL in Pasadena, USA, in 2009, and at Birmingham University, School of Computer Science, in 2012. F. Reinhart is particularly interested in dynamical system approaches to cognitive robotics, machine learning, pattern recognition and robotics in industrial applications. Since 2016, F. Reinhart is Senior Expert for Machine Learning and Data Science at Fraunhofer IEM in Paderborn.