Robotics and Autonomous Systems 205 (2026) 105649

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A context-adaptive policy framework for robust and reactive robotic manipulation via uncertainty-aware imitation learning

Tim R. Winter *, Leonard Klüpfel, Ashok M. Sundaram, Werner Friedl, Maximo A. Roa, Freek Stulp, João Silvério

German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchener Str. 20, Weßling, 82234, Germany

ARTICLE INFO

Keywords:
Imitation learning
Learning from demonstration
Machine learning for robot control
Robotic manipulation
Dynamical systems
Uncertainty awareness

ABSTRACT

Generating robust and reactive manipulation strategies that can adapt to changing context information is a challenging task in robotics. Over the years, Learning from Demonstration (LfD) has emerged as an intuitive and effective solution for generating reactive policies, particularly by following dynamical-system(DS)-based approaches. However, most state-of-the-art DS-based approaches focus on addressing the robustness limitations, overlooking the modulation of policies in response to the environment. As a result, they tend to be inflexible with respect to parameterization by task-dependent variables. In this work, we build on existing work on policy fusion and uncertainty quantification to propose a context-adaptive policy framework that combines task-parameterized, robust and reactive manipulation. For this, we use LfD to acquire a policy that is conditioned on the robot state and low-dimensional task-dependent parameters reflecting the environment. We combine the learned policy with additional uncertainty-aware policies using a Mixture of Experts (MoE) formulation to improve its out-of-distribution (OOD) robustness and convergence behavior. The approach is evaluated on the LASA handwriting dataset and on a real 7-DoF robot in three scenarios: force-conditioned grasping, manipulation of deformable food items and object-centric grasping.

1. Introduction

Context-adaptive manipulation is crucial in tasks in which the robot needs to adapt its motion to its state and task-dependent parameters. This becomes particularly relevant in the field of deformable object manipulation, as the manipulation strategy is often influenced by the objects' changeable shape, or when contact forces need to be incorporated. Due to the complexity inherent in modeling this context[1] information, the application of conventional planning and control algorithms is restricted. Consequently, recent approaches focus on solving this problem using data-driven machine learning techniques. Especially, the application of LfD approaches in the domain of robotic manipulation has gained popularity in recent years, due to the typically reduced data requirements and the adaptability to new tasks [1]. However, classical LfD approaches have focused either on time-variant or DS-based policies, both exhibiting limited reactivity properties to changing environments. An in-depth review of related work can be found in Section 2.

In this work, we take inspiration from prior achievements in policy fusion and uncertainty quantification to design a robust DS-based policy framework that can be modulated in response to the environment. For this, we propose an MoE formulation with uncertainty-aware experts and mixing coefficients, where individual expert policies are active in distinct regions of the state space. We employ Gaussian Process Regression (GPR) [2] (Section 3) due its data efficiency to learn an *LfD policy* conditioned on the robot state and task-dependent parameters. Moreover, we make use of the analytical uncertainty formulation provided by GPR to derive a *stabilizing policy* that guides the robot towards regions with low epistemic uncertainty, preventing undesired behavior, and design a kernel-based *goal attractor policy* that empirically enhances the convergence behavior of the framework. Section 4 introduces our approach. Concretely, the main contributions of this work can be summarized as:

- Incorporating the task-parameters explicitly into the learned policy (Section 4.1) — [3–6] consider only the robot state and are thus not able to adapt to the environment.
- Formulating the problem as an MoE framework with varying, uncertainty-aware mixing coefficients (Section 4.4) — works like [5,6] assume constant coefficients.

* Corresponding author.
E-mail address: tim.winter@dlr.de (T.R. Winter).

[1] In this work, we refer to context as the combination of robot state and task-dependent parameters that reflect the environment.

https://doi.org/10.1016/j.robot.2026.105649
Received 9 March 2026; Received in revised form 18 July 2026; Accepted 19 July 2026
Available online 24 July 2026

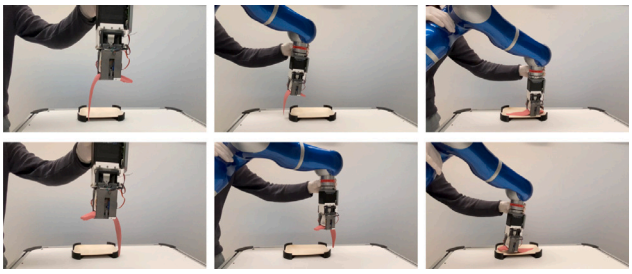


Fig. 1. In the above images we see how placing a deformable piece of silicone fish on a tray requires different approach and place strategies depending on how it has been grasped. Instead of accurately modeling the different possibilities, we propose to learn them from human demonstrations.

- Extending the stabilizing policy of [5,6] to include orientation as part of the state (Section 4.2), and introducing a novel kernel-based goal attractor policy (Section 4.3), demonstrating experimentally that both increase robustness.

Using the LASA handwriting dataset [7] and three different real-robot manipulation tasks, in Section 5 we show that our approach is able to robustly reproduce the demonstrated trajectories while adapting to changing context information, maintaining robustness when confronted with disturbances, different initial conditions or distribution shifts and reaching the demonstrated goal states. Finally, we discuss our approach in Section 6 and draw a conclusion in Section 7.

2. Related work

In literature, there is a variety of possibilities to categorize LfD approaches in robotic manipulation. In this work, we focus on approaches that learn a probabilistic policy $p_\theta(\boldsymbol{\xi}|\boldsymbol{s})$ which maps states $\boldsymbol{s}$ to actions $\boldsymbol{\xi}$ based on a demonstration dataset, where $\boldsymbol{\theta}$ are the policy parameters. The learned policy can then be executed to reproduce the demonstrated behavior at test input $\boldsymbol{s}_*$, i.e., $\boldsymbol{\xi}_* = f_\theta(\boldsymbol{s}_*)$.

Early works concentrate mostly on time-variant policies (e.g., $p_\theta(\boldsymbol{\xi}|t)$). This includes classical regression approaches applied to the domain of robotic motion generation, such as Gaussian Mixture Regression (GMR) [8,9], Gaussian Process Regression (GPR) [10], and Locally Weighted Regression (LWR) [11]. In addition, the concept of movement primitives (MP) was exploited to encode complex motor behaviors through the superposition of parameterized basis functions. Notable examples include Dynamic MP (DMP) [12], Probabilistic MP (ProMP) [13], and Kernelized MP (KMP) [14]. In order to increase generalization and adaptability, these approaches have been extended into task-parameterized or context-aware variants, which modulate the generated policies in response to environmental or task-dependent parameters. Among these, Task-Parameterized Gaussian Mixture Models (TP-GMM) [9,15], Task-Parameterized DMP (TP-DMP) [16,17] and Task-Parameterized ProMP (TP-ProMP) [18,19] have gained particular prominence, with several proposed enhancements targeting more flexible adaptation. However, these approaches impose strong assumptions on the task-parameters and, due to their time-dependent nature, exhibit limited spatial robustness and restricted capacity to adapt to dynamically changing environments.

DS-based policies, which learn a function from robot state $\boldsymbol{x}$ to its derivative $\dot{\boldsymbol{x}}$, i.e., $p_\theta(\dot{\boldsymbol{x}}|\boldsymbol{x})$, are a popular approach to address the aforementioned limitations, as they offer the capacity for real-time trajectory adaptation due to the continuous incorporation of the robot state. However, these methods often struggle to maintain robust performance under disturbances, unseen initial conditions or distribution shifts, which can lead to significant trajectory deviations or unpredictable behavior. Rather than explicitly addressing these robustness limitations, recent approaches such as diffusion policies [20] or flow-based policies [21] aim to achieve robust behavior by increasing the diversity and coverage of the training distribution. Consequently, these approaches typically require large datasets to achieve robust behavior, and inference is often computationally expensive. Nevertheless, reliable performance is generally limited to states that remain sufficiently close to the training distribution.

In contrast, several works have focused on learning stable dynamical systems with formal guarantees, including Lyapunov-constrained Gaussian mixture models [3] and stochastically stable Gaussian process state-space models [22]. More recent approaches combine expressive neural models with stability guarantees, including energy-based formulations [23], contrastive learning [24], stability-constrained deep dynamics [25], and Neural ODEs with Lyapunov and barrier constraints [26]. While these methods effectively learn stable dynamics from data, they typically restrict the permissible input and output variables to the robot state and its derivatives, which limits their adaptability to changing environments. Therefore, recent advances have integrated stability guarantees into task-parameterized frameworks, such as Elastic-DS [27] and SE(3) LPV-DS [28], improving adaptability. However, these methods — in line with the broader task-parameterized literature [9] — build on geometric or Cartesian-frame-based task parameterization, such as object poses and spatial transformations, rather than generic task-parameters, e.g., object shape descriptors, measured contact forces or task phase variables. A comprehensive overview of DS-based imitation learning, including stability-aware methods, is provided by [29].

Another line of DS-based imitation learning works tackle the generalization limitations without offering formal control-theoretic guarantees; instead they achieve robustness based on policy fusion [4–6]. Although these approaches do not impose restrictions on state and action variables and would therefore, in principle, allow modulation by arbitrary parameters, they routinely overlook the incorporation of variables different from the robot state and its derivatives, limiting their applicability in changing environments.

Thus, to the best of the authors' knowledge, a unified approach combining real-time adaptation, OOD robustness, and modulation by arbitrary external parameters remains largely underexplored. In this work, we propose to address this gap using a Mixture of Experts formulation [30] with a GPR-based policy backbone that provides uncertainty estimates. We leverage these estimates for two purposes: to design a policy that enhances OOD robustness, and to determine expert activation weights.

3. Background

In this section, we present the methodological background for the development of our approach (Section 4).

3.1. Gaussian process regression

Gaussian Process Regression (GPR) [2] is a probabilistic, non-parametric approach to regression, wherein the latent function is assumed to follow a Gaussian process prior. Given a set of N training inputs and the corresponding noisy outputs $\{\boldsymbol{s}_n, \boldsymbol{\xi}_n\}_{n=1}^N$, GPR leverages a joint Gaussian prior over the latent function values. Here, $\boldsymbol{s}_n \in \mathbb{R}^{\mathcal{I}}$ specifies the input and $\boldsymbol{\xi}_n \in \mathbb{R}^{\mathcal{O}}$ denotes the corresponding output and $\mathcal{I}$ and $\mathcal{O}$ are the dimensions of input and output space. For a multidimensional output GP with independent output variables and shared hyperparameters across the dimensions, a posterior predictive distribution for a single test input $\boldsymbol{s}_*$ can be derived by conditioning the joint prior on the observed data, providing both, the mean prediction $\boldsymbol{\mu}_* \in \mathbb{R}^{\mathcal{O}}$ and the associated predictive latent-function variance $v_* \in [0, 1]$:

$$\boldsymbol{\mu}_* = \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{\Xi}, \tag{1}$$

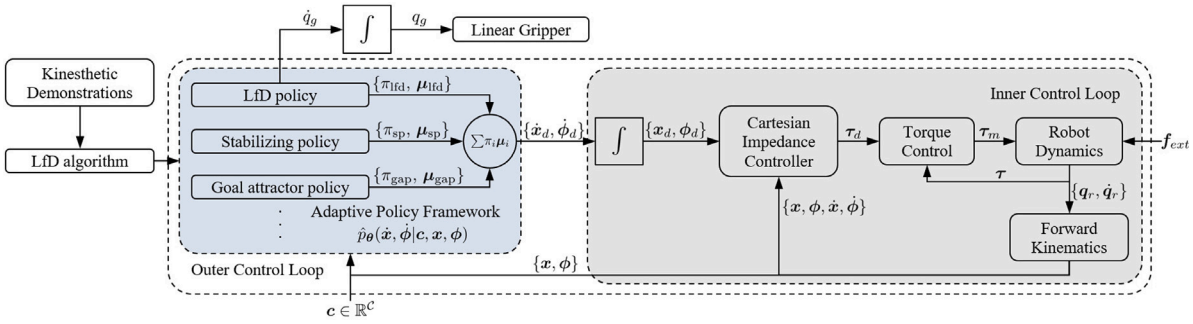


Fig. 2. Schematic overview of the two-layered control loop structure with the context-adaptive policy framework (highlighted in blue) that generates desired end-effector velocities in the outer control loop and the inner control loop (highlighted in gray) that makes the robot follow the desired velocities.

$$v_* = k_{**} - \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_*. \tag{2}$$

Here, $k_{**} = k(\boldsymbol{s}_*, \boldsymbol{s}_*)$ denotes the kernel matrix computed at test input $\boldsymbol{s}_*$, $\boldsymbol{k}_* = \begin{bmatrix} k(\boldsymbol{s}_*, \boldsymbol{s}_1) & \cdots & k(\boldsymbol{s}_*, \boldsymbol{s}_N) \end{bmatrix}^\top$ is a kernel matrix combining test input and training inputs and

$$\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{s}_1, \boldsymbol{s}_1) & \cdots & k(\boldsymbol{s}_1, \boldsymbol{s}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{s}_N, \boldsymbol{s}_1) & \cdots & k(\boldsymbol{s}_N, \boldsymbol{s}_N) \end{bmatrix} \tag{3}$$

refers to the kernel matrix evaluating the similarity of the demonstrated trajectory points. Moreover, $\boldsymbol{I}_N$ is an $N \times N$ identity matrix and $\boldsymbol{\Xi}$ is defined as $\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}_1 & \cdots & \boldsymbol{\xi}_N \end{bmatrix}^\top$.

The kernel function $k(\boldsymbol{s}_i, \boldsymbol{s}_j)$ is chosen according to the characteristics of the data. In this work, a unit-amplitude radial basis function (RBF) kernel with $k_{**} = 1$, given by

$$k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp\left(-\frac{1}{2}(\boldsymbol{s}_i - \boldsymbol{s}_j)^\top \boldsymbol{L} (\boldsymbol{s}_i - \boldsymbol{s}_j)\right), \tag{4}$$

is used as we assume smooth, continuous trajectories. The length scales $\boldsymbol{L} = \text{diag}(\boldsymbol{l})^{-2}$, with $\boldsymbol{l} = \begin{bmatrix} l_1 \cdots l_I \end{bmatrix} \in \mathbb{R}^I$, and the noise variance σ^2 denote the hyperparameters of GPR.

The predictive variance of GPR (Eq. (2)) provides a closed-form epistemic uncertainty[2] quantification, that increases with the distance to the training data, and therefore helps to identify out of distribution states.

3.2. Probabilistic mixture of experts

The combination of expert models is a common machine learning technique for modeling high-dimensional data that adheres to multiple low-dimensional constraints. It facilitates learning or formulating several simpler probability distributions that cover each a low-dimensional constraint and combining their outputs. The two most common methods are the Mixture of Experts (MoE) [30] and the Product of Experts (PoE) [32]. With PoE, all conditions must be approximately satisfied to obtain the desired output, and it thus corresponds to an "and" operation. MoE can be interpreted as an "or" operation that returns a weighted arithmetic mean of the individual expert distributions.

Unlike prior work that uses PoE to jointly satisfy multiple simultaneously active objectives within a shared task execution space [33] or employs gating mechanisms to select a single conditional expert model for representing multimodal movement primitives [34], our approach focuses on smoothly composing locally specialized policies that dominate in distinct regions of the task or parameter space. Concretely, we model the probabilistic control actions as a Gaussian distribution, i.e., $\hat{\boldsymbol{\xi}}_* \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$, whose mean and covariance are obtained via moment-matching from an MoE. The MoE is defined as $\hat{p}_\theta(\boldsymbol{\xi}|\boldsymbol{s}) = \sum_{i=1}^P \pi_i p_{\theta,i}(\boldsymbol{\xi}|\boldsymbol{s})$, where P is the number of experts, $p_{\theta,i}(\boldsymbol{\xi}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ represents the probability density function of expert i and π_i is its mixing coefficient, with $\sum_{i=1}^P \pi_i = 1$. The resulting mixture moments are given by

$$\hat{\boldsymbol{\mu}} = \sum_{i=1}^P \pi_i \boldsymbol{\mu}_i, \quad \hat{\boldsymbol{\Sigma}} = \sum_{i=1}^P \pi_i \left(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top\right) - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^\top. \tag{5}$$

4. Context-adaptive policy framework

The aim of the context-adaptive policy framework is to provide a general framework for robust and reactive robotic manipulation within dynamically changing environments. In order to achieve the reactive behavior of the robotic system, we apply a two-layer control loop structure, displayed in Fig. 2, as the basis of our approach. Here, the inner control loop (highlighted in gray), consisting of a low-level controller and the robot's dynamics, makes the robot follow desired linear and angular end-effector velocities $\dot{\boldsymbol{x}}_d$, $\dot{\boldsymbol{\phi}}_d$.[3] These velocities are obtained in the outer control loop[4] by querying a context-adaptive policy $\hat{p}_\theta(\dot{\boldsymbol{x}}, \dot{\boldsymbol{\phi}} | \boldsymbol{c}, \boldsymbol{x}, \boldsymbol{\phi})$ at each time step. This DS-based policy is conditioned on the end-effector pose, defined by position $\boldsymbol{x}$ and orientation $\boldsymbol{\phi}$, and unlike state-of-the-art stable DS-based learning frameworks, also on generic low-dimensional task-dependent parameters $\boldsymbol{c} \in \mathbb{R}^C$, capturing task-relevant environmental features such as contact forces (Section 5.3), object shape descriptors (Section 5.4) or phase variables (Section 5.6). The outer control loop is exited if the robot has remained close to the goal pose for a predefined number of steps, a maximum number of iterations is reached, or an external stop is triggered.

Since, in the case of a purely data-driven policy $\hat{p}_\theta$, the described control structure would lead to unpredictable robot behavior when $\boldsymbol{x}$, $\boldsymbol{\phi}$ or $\boldsymbol{c}$ deviate from the trained distribution, we propose to formulate the context-adaptive policy as an MoE that combines an LfD policy with supplementary policies that keep the robot in low epistemic uncertainty regions and empirically enhance the convergence behavior. While DS-based approaches that enforce constraints (e.g., stability [3]) on the LfD policy typically restrict inputs and outputs to the robot state and its derivatives, the combination of multiple policies enables the incorporation of additional, non-robot-related inputs into the LfD policy. The proposed MoE formulation thus facilitates the modulation by arbitrary external low-dimensional task-dependent parameters while enabling robust and reactive manipulation. In addition, the modular structure of our MoE formulation allows policies to be added and removed ad-hoc, contributing to better explainability and interpretability.

[2] Epistemic uncertainty arises from limited knowledge of the model and can be reduced with more data, whereas aleatoric uncertainty reflects inherent stochasticity in the data; see [31] for an introduction.

[3] In this manuscript, we assume task-space control, which enables embodiment-independent motion specification and improves interpretability of the supplementary policies. However, a joint-space implementation would also be possible in principle.

[4] Note that the control rate of the outer control loop can be lower than the rate of the inner control loop, which is 1 kHz in this work.

As backbone of the context-adaptive policy framework (blue part in Fig. 2), we propose three distinct policies,[5] which we treat as Gaussian-distributed experts with means $\boldsymbol{\mu}_i$ and mixing coefficients π_i (Eq. (5)):

- **LfD policy (π_{lfd}, $\boldsymbol{\mu}_{\text{lfd}}$):** A data-driven policy that is modulated in response to the current state of the robot and task-dependent parameters, which imitates the demonstrated motions.
- **Stabilizing policy (π_{sp}, $\boldsymbol{\mu}_{\text{sp}}$):** An uncertainty-aware policy that guides the robot towards regions with low epistemic uncertainty, overcoming the covariate shift and thus maintaining reliable performance.
- **Goal attractor policy (π_{gap}, $\boldsymbol{\mu}_{\text{gap}}$):** A kernel-based policy that empirically improves the convergence behavior of the framework.

We define the control action $\hat{\boldsymbol{\xi}}_* = \begin{bmatrix}\dot{\boldsymbol{x}}_d^\top & \dot{\boldsymbol{\phi}}_d^\top\end{bmatrix}^\top$ at test input $\boldsymbol{s}_*$ using the first moment of the MoE (Eq. (5)), resulting in:

$$\hat{\boldsymbol{\xi}}_* = \hat{\boldsymbol{\mu}} = \pi_{\text{lfd}}\boldsymbol{\mu}_{\text{lfd}} + \pi_{\text{sp}}\boldsymbol{\mu}_{\text{sp}} + \pi_{\text{gap}}\boldsymbol{\mu}_{\text{gap}}. \tag{6}$$

In the following sections we describe the formulation of the means and mixing coefficients of the individual experts.

4.1. LfD policy

In this work, we use GPR (see Section 3) to generate the LfD policy due to the small amount of data required and the explicit uncertainty formulation provided. This makes it an ideal base policy for demonstrating the benefits of our uncertainty-aware framework. Since GPs are computationally limited in the number of training data they can handle efficiently, we subsample N points from a set of H demonstrated trajectories $\{\{\boldsymbol{s}_{m,h}, \boldsymbol{\xi}_{m,h}\}_{m=1}^{M_h}\}_{h=1}^{H}$, with M_h samples. The subsampled training inputs $\boldsymbol{s}_n = \begin{bmatrix}\boldsymbol{c}_n^\top & \boldsymbol{x}_n^\top & \boldsymbol{\phi}_n^\top\end{bmatrix}^\top$ consist of the end-effector pose, along with additional low-dimensional task-dependent parameters (see Section 4), while the corresponding actions $\boldsymbol{\xi}_n = \begin{bmatrix}\dot{\boldsymbol{x}}_n^\top & \dot{\boldsymbol{\phi}}_n^\top\end{bmatrix}^\top$ contain the linear and angular end-effector velocities, as typical in DS-based learning frameworks (e.g., [3]). Since the posterior predictive distribution exhibited by GPR already fulfills the requirement to be a Gaussian-distributed expert, we define $\boldsymbol{\mu}_{\text{lfd}} = \begin{bmatrix}\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^\top & \boldsymbol{\mu}_{\dot{\boldsymbol{\phi}}}^\top\end{bmatrix}^\top$ as the mean end-effector velocity computed by Eq. (1).[6]

The vector fields of the LfD policy are exemplarily illustrated in Fig. 3(a) for the first letter of the LASA handwriting dataset [7], where we only consider $\boldsymbol{s} = \boldsymbol{x} \in \mathbb{R}^2$, i.e., no task-dependent parameter $\boldsymbol{c}$. The false attractors displayed in the illustration (bottom-left and bottom-right regions) highlight the generalization issues of purely data-driven machine learning approaches applied to control, which lead to undesired behavior in underrepresented regions. This reaffirms the need for supplementary policies that empirically improve the OOD robustness and convergence behavior.

4.2. Stabilizing policy

Due to the aforementioned OOD generalization limitations, LfD policies typically fail to generalize beyond well-covered regions, effectively restricting operation to the vicinity of the demonstrated trajectories. To improve robustness in OOD states — i.e., states that lie beyond the support of the observed demonstrations — we introduce a stabilizing policy that actively guides the robot toward regions of low epistemic uncertainty, which typically coincide with the regions covered by the demonstration data. For the formulation of this policy, we draw upon prior work [5,6] and leverage the predictive variance of GPR (Eq. (2)), which provides a closed-form epistemic uncertainty quantification that increases smoothly with distance from the training data. Specifically, the approach generates actions along the gradient of the predictive variance, thereby directing the system toward the nearest low-uncertainty region. Additionally, we make use of the combined kernel for position and orientation, and incorporate, contrary to [5,6], orientation in the stabilizing policy. Since the state space $\boldsymbol{s}_n = \begin{bmatrix}\boldsymbol{c}_n^\top & \boldsymbol{x}_n^\top & \boldsymbol{\phi}_n^\top\end{bmatrix}^\top$ jointly encodes both Cartesian position and end-effector orientation, the RBF kernel naturally operates on this full pose representation. The predictive variance and its gradient therefore inherently capture uncertainty contributions from both translation and rotation, enabling the stabilizing policy to simultaneously steer the robot in position and orientation.

For this, we derive the gradient of v_* with respect to the current state $\boldsymbol{s}_*$,[7] assuming the RBF kernel Eq. (4):

$$\nabla v_* = -2\nabla \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_*, \tag{7}$$

where $\nabla \boldsymbol{k}_* = \begin{bmatrix}\nabla k(\boldsymbol{s}_*, \boldsymbol{s}_1)^\top & \cdots & \nabla k(\boldsymbol{s}_*, \boldsymbol{s}_N)^\top\end{bmatrix}^\top$ and $\nabla k(\boldsymbol{s}_*, \boldsymbol{s}_n) = -k(\boldsymbol{s}_*, \boldsymbol{s}_n)\boldsymbol{L}(\boldsymbol{s}_* - \boldsymbol{s}_n)$ is the gradient of the RBF kernel. In this work we assume that, in contrast to the robot state, the task-dependent parameter $\boldsymbol{c}$ can only be controlled indirectly and thus focus on the robot state dependent terms $\nabla_{\boldsymbol{x}_*} v_*$, $\nabla_{\boldsymbol{\phi}_*} v_*$ to design the stabilizing policy.[8] We normalize the gradients for position and orientation independently to unit length and introduce the hyperparameter $\boldsymbol{K}_{\text{sp}} = \{K_{\text{sp},\boldsymbol{x}}, K_{\text{sp},\boldsymbol{\phi}}\}$ to map the normalized gradients to the velocity domain. Moreover, to regulate the magnitude of the velocities, we scale them by the respective predictive variance magnitudes $v_{\boldsymbol{c},\boldsymbol{x}}$, $v_{\boldsymbol{c},\boldsymbol{\phi}}$, which are computed using Eq. (2) with $\boldsymbol{s} = \begin{bmatrix}\boldsymbol{c}^\top & \boldsymbol{x}^\top\end{bmatrix}^\top$ and $\boldsymbol{s} = \begin{bmatrix}\boldsymbol{c}^\top & \boldsymbol{\phi}^\top\end{bmatrix}^\top$, respectively. This leads to vanishing velocities towards the associated demonstrated trajectories. Based on this, we define the mean of the stabilizing policy distribution as:

$$\boldsymbol{\mu}_{\text{sp}} = \begin{bmatrix} -K_{\text{sp},\boldsymbol{x}} \dfrac{\nabla_{\boldsymbol{x}_*} v_*}{\max(\|\nabla_{\boldsymbol{x}_*} v_*\|, \varepsilon)} v_{\boldsymbol{c},\boldsymbol{x}} \\ -K_{\text{sp},\boldsymbol{\phi}} \dfrac{\nabla_{\boldsymbol{\phi}_*} v_*}{\max(\|\nabla_{\boldsymbol{\phi}_*} v_*\|, \varepsilon)} v_{\boldsymbol{c},\boldsymbol{\phi}} \end{bmatrix}, \tag{8}$$

where ε is a small positive constant to prevent numerical issues.[9] Eq. (8) guides the robot towards regions with low epistemic uncertainty. This effect can be observed on the vector fields in Fig. 3(b).

4.3. Goal attractor policy

GPR alone is not guaranteed to converge to the demonstrated goal poses $\boldsymbol{s}_{M_h,h}$, as can be seen in Fig. 3(a). While the stabilizing policy improves OOD robustness and safety, it does not explicitly enforce the robot to reach and maintain the goal pose. To address this, we further introduce a goal-attractor policy that steers the robot toward the associated goal pose $\boldsymbol{s}_g = \text{argmax}\{k(\boldsymbol{s}_*, \boldsymbol{s}_{M_1,1}), \ldots, k(\boldsymbol{s}_*, \boldsymbol{s}_{M_H,H})\}$, which we define as the demonstrated goal pose with the highest kernel activation at current state $\boldsymbol{s}_*$. As the RBF kernel activation quantifies the similarity of two given states, $\boldsymbol{s}_g$ corresponds to the demonstrated goal pose with the highest similarity to the current state in the combined feature space. To guide the robot toward $\boldsymbol{s}_g$, we adopt the natural choice of following the gradient w.r.t. the robot state of this activation, given by $\nabla k(\boldsymbol{s}_*, \boldsymbol{s}_g)$ with $\boldsymbol{s}_g = \begin{bmatrix}\boldsymbol{c}_g^\top & \boldsymbol{x}_g^\top & \boldsymbol{\phi}_g^\top\end{bmatrix}^\top$, i.e., to move in the direction that increases the kernel value.

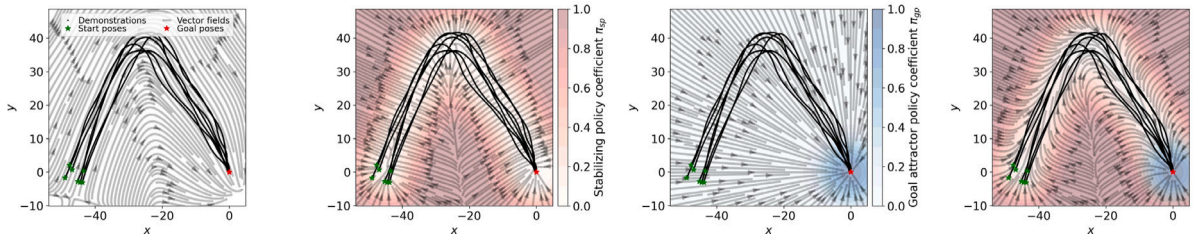
[5] Additional policies can, in principle, be included.

[6] In the experiments, we extend the output $\boldsymbol{\xi}$ of the LfD policy by the joint angle velocity of the gripper $\dot{q}_g$ in order to infer the grasp behavior as well. The mean computed according to Eq. (1) therefore contains three distinct terms $\boldsymbol{\mu}_* = \begin{bmatrix}\boldsymbol{\mu}_{\dot{\boldsymbol{x}}}^\top & \boldsymbol{\mu}_{\dot{\boldsymbol{\phi}}}^\top & \mu_{\dot{q}_g}^\top\end{bmatrix}^\top$. Within the MoE formulation, we still consider only the mean end-effector velocity, while the mean joint angle velocity $\mu_{\dot{q}_g}$ is used separately to control the gripper as can be seen in Fig. 2.

[7] For simplicity and improved readability, we write ∇ instead of $\nabla_{\boldsymbol{s}_*}$.

[8] Note that for an input of the form of $\boldsymbol{s}$, the gradient contains three distinct terms, $\nabla v_* = \begin{bmatrix}(\nabla_{\boldsymbol{c}_*} v_*)^\top & (\nabla_{\boldsymbol{x}_*} v_*)^\top & (\nabla_{\boldsymbol{\phi}_*} v_*)^\top\end{bmatrix}^\top$.

[9] In practice, we set it to the smallest positive floating-point number available in Python, 2.2×10^{-308}.



(a) LfD policy (b) Stabilizing policy (c) Goal attractor policy (d) MoE

Fig. 3. Two-dimensional example showing the vector fields and the mixing coefficient activation of the MoE and the individual policies when trained on the first letter of the LASA handwriting dataset ('Angle'). (a) shows the LfD policy, (b) illustrates the stabilizing policy, (c) displays the goal attractor policy and (d) demonstrates the combination of the individual policies using the MoE formulation.

As with the stabilizing policy, we normalize the gradients for position and orientation independently and introduce the hyperparameter $\boldsymbol{K}_{\text{gap}} = \{K_{\text{gap},\boldsymbol{x}}, K_{\text{gap},\boldsymbol{\phi}}\}$ to map the normalized gradients to the velocity domain. But, in contrast, we now use the kernels $k(\boldsymbol{x}_*, \boldsymbol{x}_g)$ and $k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_g)$ to obtain vanishing velocities towards the goal pose, and retain the positive gradient as we wish to increase the kernel activation. Consequently, we define the mean of the goal attractor policy distribution as:

$$\boldsymbol{\mu}_{\text{gap}} = \begin{bmatrix} K_{\text{gap},\boldsymbol{x}} \dfrac{\nabla_{\boldsymbol{x}_*} k(\boldsymbol{s}_*, \boldsymbol{s}_g)}{\max(\|\nabla_{\boldsymbol{x}_*} k(\boldsymbol{s}_*, \boldsymbol{s}_g)\|, \epsilon)} (1 - k(\boldsymbol{x}_*, \boldsymbol{x}_g)) \\ K_{\text{gap},\boldsymbol{\phi}} \dfrac{\nabla_{\boldsymbol{\phi}_*} k(\boldsymbol{s}_*, \boldsymbol{s}_g)}{\max(\|\nabla_{\boldsymbol{\phi}_*} k(\boldsymbol{s}_*, \boldsymbol{s}_g)\|, \epsilon)} (1 - k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_g)) \end{bmatrix}. \tag{9}$$

The vector fields resulting from Eq. (9) are visualized in Fig. 3(c) for the considered illustrative example.

4.4. MoE mixing coefficient design

We here outline the design choices for the mixing coefficients of Eq. (6) that are automatically computed based on the uncertainty inherent in the provided demonstrations.

4.4.1. Goal attractor policy coefficient π_{gap}

Close to the respective goal, the robot should approach it and maintain its pose there. Consequently, the activation is designed to increase as the distance to the goal pose decreases. To facilitate this local activation and smooth transitions between the policies, we define

$$\pi_{\text{gap}} = k(\boldsymbol{s}_*, \boldsymbol{s}_g) \tag{10}$$

as the goal attractor policy coefficient, where $\boldsymbol{s}_g$ is recomputed at each iteration based on the current state $\boldsymbol{s}_*$.

4.4.2. Stabilizing policy coefficient π_{sp}

Furthermore, we assign a high priority to staying close to demonstrations, avoiding dangerous, unseen behaviors. Therefore, we use the variance prediction provided by GPR (Eq. (2)), which increases smoothly when deviating from the data, as activation. Unlike approaches with constant coefficients [5,6], this yields a state-dependent activation that adapts according to the predictive uncertainty.

We reduce the mixing coefficient by the activation of the goal attractor policy, as we assign a higher priority to it close to the goal. We thus define

$$\pi_{\text{sp}} = (1 - \pi_{\text{gap}}) v_* = (1 - k(\boldsymbol{s}_*, \boldsymbol{s}_g)) v_*. \tag{11}$$

4.4.3. LfD policy coefficient π_{lfd}

Taking into account $\sum_{i=1}^{P} \pi_i = 1$ (Section 3.2) results in

$$\pi_{\text{lfd}} = 1 - \pi_{\text{gap}} - \pi_{\text{sp}} = (1 - k(\boldsymbol{s}_*, \boldsymbol{s}_g))(1 - v_*) \tag{12}$$

as the mixing coefficient for the LfD policy. Hence, the LfD policy is smoothly activated near the demonstrations, while the goal attractor policy takes over close to the goal.

The mixing coefficients for each policy as well as the combined MoE output $\hat{\boldsymbol{\mu}}$, applied to the 2D example, are displayed in Fig. 3 using contour lines. Here, each policy acts as an expert in its respective area, while the combined policy unifies the advantages of the individual policies.

Algorithm 1 summarizes this section in pseudo-code.[10]

Algorithm 1 Skill learning and execution

Input: $\{\{\boldsymbol{s}_{m,h}, \boldsymbol{\xi}_{m,h}\}_{m=1}^{M_h}\}_{h=1}^{H}$, $\{\boldsymbol{I}, \sigma^2, \boldsymbol{K}_{\text{sp}}, \boldsymbol{K}_{\text{gap}}\}$, $\boldsymbol{s}_0$

$\{\boldsymbol{s}_n, \boldsymbol{\xi}_n\}_{n=1}^{N} \subset \{\{\boldsymbol{s}_{m,h}, \boldsymbol{\xi}_{m,h}\}_{m=1}^{M_h}\}_{h=1}^{H}$ ▷ Subsample data

$\boldsymbol{R} \leftarrow \text{chol}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})$ ▷ Train GP (cf. [2], Alg. 2.1)

$\boldsymbol{\alpha} \leftarrow \boldsymbol{R}^{\top} \backslash (\boldsymbol{R} \backslash \boldsymbol{\Xi})$

$\boldsymbol{s}_* \leftarrow \boldsymbol{s}_0$ ▷ Assign initial conditions

$i \leftarrow 0$ ▷ Reset iteration counter

goalReached $\leftarrow$ false

stopTriggered $\leftarrow$ false

while $\neg$ goalReached $\wedge$ $i < I_{\max}$ $\wedge$ $\neg$ stopTriggered **do**

  $\boldsymbol{\mu}_{\text{lfd}} \leftarrow$ Eq. (1) ▷ Compute GP mean ([2], Alg. 2.1)

  $\boldsymbol{\mu}_{\text{sp}} \leftarrow$ Eq. (8) ▷ Calculate stab. policy mean

  $\boldsymbol{\mu}_{\text{gap}} \leftarrow$ Eq. (9) ▷ Determine goal attr. policy mean

  $\pi_{\text{gap}} \leftarrow$ Eq. (10) ▷ Calculate goal attr. mixing coeff.

  $\pi_{\text{sp}} \leftarrow$ Eq. (11) ▷ Determine stab. mixing coeff.

  $\pi_{\text{lfd}} \leftarrow$ Eq. (12) ▷ Derive LfD mixing coeff.

  $\hat{\boldsymbol{\xi}}_* \leftarrow$ Eq. (6) ▷ Compute control action (MoE mean)

  $\boldsymbol{s}_* \leftarrow$ InnerControlLoop($\hat{\boldsymbol{\xi}}_*$) ▷ Apply $\hat{\boldsymbol{\xi}}_*$ to robot

  goalReached $\leftarrow$ checkGoalReached($\boldsymbol{s}_*$)

  stopTriggered $\leftarrow$ checkExternalStop()

  $i \leftarrow i + 1$ ▷ Increment iteration counter

end while

5. Evaluation

We evaluate our approach in four experiments of increasing complexity. First, we investigate its performance on the LASA handwriting dataset [7], omitting the task-dependent parameter $\boldsymbol{c}$, and focusing on its OOD robustness and convergence behavior with only the robot state as input (Section 5.2). We then use a real 7-DoF robot in three manipulation scenarios, namely force-conditioned grasping (Section 5.3), manipulation of deformable food items (Section 5.4) and object-centric grasping (Section 5.6).[11] For these scenarios, task selection was guided by the aim to consider diverse contextual information, realized through the incorporation of both continuous and discrete task-dependent parameters such as contact forces (Section 5.3), object shape descriptors (Section 5.4) and phase variables (Section 5.6). The hyperparameter values used are obtained through black box optimization as described in Section 5.1 and are listed in Table A.1.

[10] A definition of *goalReached* is given in Section 5.1.

[11] Videos of all experiments can be found as supplementary material.

Table 1
Quantitative evaluation (average success S, iterations I, distance D) on the LASA handwriting dataset. Results are averaged over all letters and reported as the mean and standard deviation across 20 random seeds. Each seed includes multiple trials per letter: 10 with random initialization ("$\boldsymbol{x}_{0,k}$ random") and 7 with on-track initialization (first point in each of the 7 demonstrations, "$\boldsymbol{x}_{0,k}$ on demos"). **Note:** Since actions are generated from the first moment of the MoE, our policy is deterministic for a fixed set of hyperparameters. Thus, variation across evaluation seeds stems solely from the initial poses. When starting on demonstrations, the initial poses for each letter are the same across all seeds, resulting in standard deviations of 0.0.

Exp.	Method	S [%]	I	D
$\boldsymbol{x}_{0,k}$ on demos	GPR (LfD policy 4.1, [2])	11.0 ± 0.0	454.9 ± 0.0	2.11 ± 0.00
	GPR & stab. (4.2)	86.7 ± 0.0	165.8 ± 0.0	0.23 ± 0.00
	GPR & goal attr. (4.3)	99.5 ± 0.0	$\mathbf{113.3 \pm 0.0}$	0.36 ± 0.00
	Proposed framework	$\mathbf{100.0 \pm 0.0}$	114.3 ± 0.0	$\mathbf{0.20 \pm 0.00}$
	Diffusion Policy ([20])	9.4 ± 1.0	477.5 ± 2.0	43.61 ± 1.96
	SEDS (likelihood, [3])	99.8 ± 0.3	125.8 ± 3.9	2.99 ± 1.29
$\boldsymbol{x}_{0,k}$ random	GPR (LfD policy 4.1, [2])	5.1 ± 1.0	480.5 ± 3.8	8.07 ± 0.29
	GPR & stab. (4.2)	86.7 ± 0.0	127.5 ± 1.7	$\mathbf{0.65 \pm 0.03}$
	GPR & goal attr. (4.3)	58.6 ± 2.2	270.1 ± 9.0	7.40 ± 0.30
	Proposed framework	$\mathbf{100.0 \pm 0.0}$	$\mathbf{72.0 \pm 1.7}$	$\mathbf{0.65 \pm 0.03}$
	Diffusion Policy ([20])	10.9 ± 1.1	470.7 ± 4.2	60.31 ± 4.38
	SEDS (likelihood, [3])	99.5 ± 0.7	100.9 ± 5.6	3.52 ± 0.99

5.1. Hyperparameter optimization

Given the influence of GPR hyperparameters on the supplementary policies (e.g., through the shared kernel length over all policies), the optimization of the combined hyperparameters $\boldsymbol{\theta} = \{l, \sigma^2, \boldsymbol{K}_{\text{sp}}, \boldsymbol{K}_{\text{gap}}\}$ is performed jointly using the CMA-ES algorithm [35], which is used due to its derivative-free nature and its ability to efficiently search complex parameter spaces. To this end, the skill learning and execution procedure, as summarized in Algorithm 1, is executed ten times for the provided demonstrations, while the robot behavior (*InnerControlLoop*) is simulated by computing the next pose at each time step using $\boldsymbol{x}_{t+1,i} = \boldsymbol{x}_{t,i} + \Delta t \dot{\boldsymbol{x}}_{t,i}$ and $\boldsymbol{\phi}_{t+1,i} = \boldsymbol{\phi}_{t,i} + \Delta t \dot{\boldsymbol{\phi}}_{t,i}$. To optimize performance both on and off the demonstrated trajectories, the starting poses $\{\boldsymbol{x}_{0,k}, \boldsymbol{\phi}_{0,k}\}_{k=1}^{10}$ are sampled randomly within the demonstrated input space, naturally accounting for sensitivity by evaluating performance across diverse starting poses. In cases where task-dependent parameters are employed, the recorded values $\{\boldsymbol{c}_{m,h}\}_{m=1}^{M_h}$ from a random demonstration h are re-injected into the policy.

Each of the ten simulated trials is terminated when the robot stayed in the goal region for ten iterations (*goalReached*) or $I_{\max} = 500$ iterations are reached. Defining $\boldsymbol{p} = \left[\boldsymbol{x}^\top \boldsymbol{\phi}^\top\right]^\top \in \mathbb{R}^{\mathcal{P}}$ as the end-effector pose, the robot has reached the goal region if each component $p_{*,r}$ of the current end-effector pose $\boldsymbol{p}_*$ lies within the distance $d_r = 0.01 \cdot d_{\max,r}$ to one of the demonstrated goal poses $\{\boldsymbol{p}_{M_h,h}\}_{h=1}^{H}$, with $d_{\max,r} = \max_{m,h} p_{m,h,r} - \min_{m,h} p_{m,h,r}$ describing the max-min range across all demonstrations for the specific entry and $r \in [0, \mathcal{P}]$. We classify trials in which the robot successfully remained in the goal region as successful $S_k = 1$; otherwise, they are classified as unsuccessful $S_k = 0$. Further, we measure the number of iterations $i_k \in [0, I_{\max}]$ and the mean distance between the executed poses and the closest demonstrated poses across all time steps $D_k = \frac{1}{i_k} \sum_{j=0}^{i_k} \min_{m,h} \left\| \boldsymbol{p}_{m,h} - \boldsymbol{p}_j \right\|$. We define the multi-objective cost function as an equally weighted sum:

$$\min_{\boldsymbol{\theta}} J = (1 - S) + \frac{I}{I_{\max}} + \frac{D}{\sqrt{\sum_{r=1}^{\mathcal{P}} d_{\max,r}^2}}, \tag{13}$$

where, S, I and D are the average success, iterations and distance over ten trials, thus all objectives lie within $[0, 1]$. We further constrain the search space of the hyperparameters in $\boldsymbol{\theta}$ to 1% - 100% of their respective max-min ranges.

The hyperparameters for the LASA handwriting dataset experiment (Section 5.2) are obtained within 500 optimization iterations using a combined objective function and a single set of hyperparameters for all letters. As the robot state and action spaces remain the same, we optimize the hyperparameters for the real-robot experiments (Sections 5.3, 5.4 and 5.6) over 500 optimization iterations using the training data of the experiment described in Section 5.6 and then manually refined the task-parameter-dependent kernel length for the experiments described in Sections 5.3 and 5.4, as only the domain of $\boldsymbol{c}$ varies across tasks. Starting from the CMA-ES-derived initial value (Section 5.6), $l_{\boldsymbol{c}}$ was tuned via a unit-step search within the bounds of the demonstrated $\boldsymbol{c}$ range. Each candidate value was evaluated over five independent trials initialized at the demonstrated start poses. The parameter was incremented until entering, and subsequently leaving, the region of 100% success rate, thereby identifying the contiguous interval of tested values achieving perfect success. The final value of $l_{\boldsymbol{c}}$ was chosen as the median of this interval. This criterion emphasizes operational robustness and is consistent with our sensitivity analysis (Appendix C), which indicates approximately symmetric performance degradation around the unperturbed value, confirming robustness against $\boldsymbol{c}$ perturbations up to ± 2.5 N. We selected this strategy as it provides a fast and practical solution, while a direct optimization of the hyperparameters, as for 5.6, would also be possible.

5.2. LASA handwriting dataset

First, we quantitatively evaluate our approach on the LASA handwriting dataset [7] and compare it with different combinations of the individual policies[12] and two established baseline approaches: SEDS [3] and Diffusion Policy [20]. In this experiment, we focus on the reproduction of the demonstrated handwriting motions with different initial conditions, showing the approach's OOD robustness and empirical convergence behavior. Thus, we define $\boldsymbol{s} = \boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{\xi} = \dot{\boldsymbol{x}} \in \mathbb{R}^2$, omitting the task-dependent parameter $\boldsymbol{c}$. We perform two sets of simulations, similar to those described in Section 5.1, for all letters: one in which $\boldsymbol{x}_{0,k}$ is placed at the initial pose of each of the seven demonstrations and one in which $\boldsymbol{x}_{0,k}$ is placed ten times randomly within the demonstrated input ranges. As evaluation metrics, we use S, I and D (see Section 5.1). For Diffusion Policy, the *state-based environment*[13] is used, and the *likelihood* criterion is applied for SEDS [7]. The provided source code is adapted to the dataset and experimental setup, while the hyperparameters are kept at their default values — SEDS hyperparameters are already tuned for this task, and Diffusion Policy hyperparameters follow commonly employed configurations for low-dimensional control tasks. Comprehensive Diffusion Policy tuning is omitted due to the substantially higher computational effort required for both training and inference compared to our method. This disparity is reflected in the significantly higher inference time of Diffusion Policy (see Table B.3).

Table 1 shows the averaged results over 20 seeds, each comprising multiple trials (ten for random starting and seven for on-track starting) per letter, and Fig. 4 illustrates the executed trajectories, vector fields, and mixing coefficients of the MoE for a single seed across all letters of the LASA handwriting dataset, where $\boldsymbol{x}_{0,k}$ is randomly initialized ten times within the demonstrated input ranges. For completeness, the corresponding results for the same seed, with $\boldsymbol{x}_{0,k}$ initialized at the starting pose of each of the seven demonstrations, are shown in Fig. A.2.

[12] Notably, this evaluation explicitly includes GPR as a standalone LfD policy, which serves as a small-data behavioral cloning baseline.

[13] Available at https://github.com/real-stanford/diffusion_policy.

Fig. 4. Reproductions for all letters of the LASA dataset when $\boldsymbol{x}_{0,k}$ is placed ten times randomly within the demonstrated input ranges showing the executed paths, vector fields and mixing coefficients of the MoE (refer to Fig. 3 for clarification).

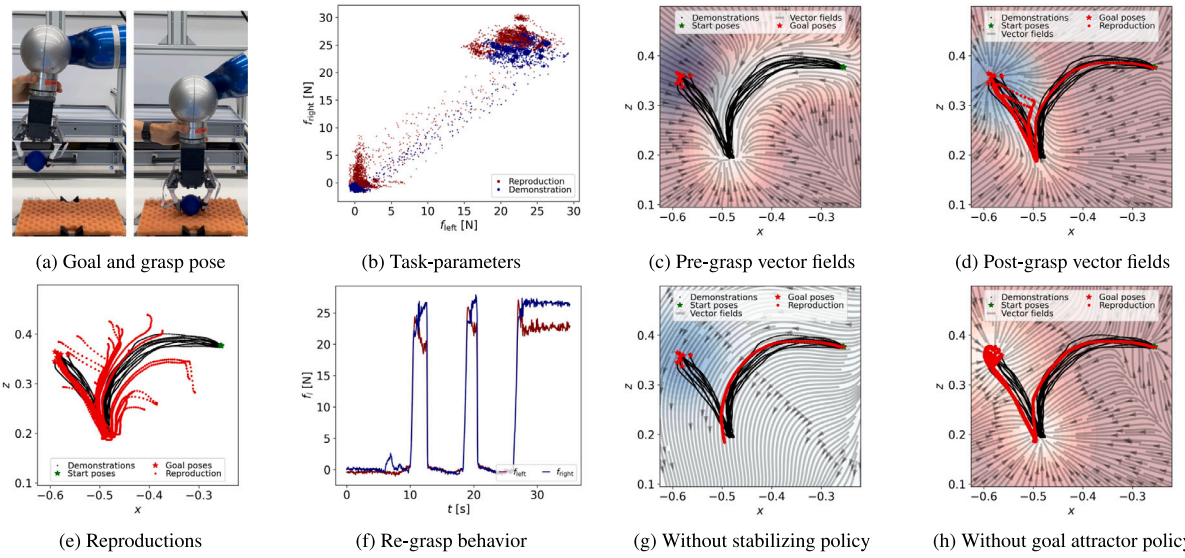
5.3. Force-conditioned grasping

In the second experiment, we demonstrate our approach's robust transferability to real robots and its ability to react to changing task-parameter values at runtime. For this, we record ten trajectories via kinesthetic teaching using a 7-DoF KUKA LWR robotic arm equipped with a variable stiffness parallel gripper, where the robot picks up a 3D-printed sphere from a table and transports it to a demonstrated goal pose (Fig. 5(a)). We make use of the incorporated tactile sensors in both fingertips to derive the forces exerted by the object and define $\boldsymbol{c} = \left[f_{\text{left}}\ f_{\text{right}}\right]^{\top}$ as task-parameter. In addition, the input includes the robot position and orientation as Euler vector, i.e., $\mathcal{I} = 8$. The demonstrated and reproduced task-parameters can be seen in Fig. 5(b).

Similar to the experiments in Section 5.2, we quantitatively evaluate our approach and different combinations of the individual policies on this task and compare it with Diffusion Policy [20].[14] For this, we perform 20 trials, where the initial poses $\{\boldsymbol{x}_{0,k}, \boldsymbol{\phi}_{0,k}\}$ are drawn uniformly within the demonstrated input ranges, as shown in Fig. 5(e). Our context-adaptive policy, whose vector fields are exemplarily displayed for a single trial at $\boldsymbol{s}_0$ (Fig. 5(c)) and $\boldsymbol{s}_I$ (Fig. 5(d)), is used to compute the end-effector velocity $\dot{\boldsymbol{x}}_d$, $\dot{\boldsymbol{\phi}}_d$ and the gripper joint velocity $\dot{q}_g$ in a 20 Hz loop, hence $\mathcal{O} = 7$. Due to the longer task horizon compared to Section 5.2, we define $I_{\max} = 1000$, and add a 1 mm or 1° offset to d_r, depending on whether it represents position or orientation, to account for measurement inaccuracies. Since this is a real-world task, in addition to S, I and D, we use the number of collisions (*Coll.*) as comparison metric, which measures all contact events including unintentional collisions with the target object, collisions with the surrounding environment, and self-collisions of the robot. Collisions are automatically detected based on joint configurations (self-collisions) and torque measurements; however, to prevent damage to the robot and gripper, the authors proactively stopped the robot if hard collisions were to be expected. Note that such events would have inevitably led to collision events detected via torque readings. The results are displayed in the upper part of Table 2 and Fig. 5(e).

Furthermore, we demonstrate the approach's reactivity by opening the gripper manually such that the object drops from the gripper. Due to the task-parameter conditioning, the robot attempts to re-grasp the sphere at its original place where it is pulled back via a cable. The resulting task-parameters are exemplified in Fig. 5(f) for the case where the gripper is opened twice. The corresponding trajectory is displayed in addition to the vector fields in Fig. 5(d). Moreover, we show that the proposed combination of policies is essential for accurate reproduction by removing either the stabilizing (Fig. 5(g)) or the goal attractor policy (Fig. 5(h)).

[14] A comparison with SEDS, as in Section 5.2, is not feasible in this and the subsequent experiments, since SEDS is restricted to specific input and output variables due to the incorporation of stability constraints.



(a) Goal and grasp pose (b) Task-parameters (c) Pre-grasp vector fields (d) Post-grasp vector fields

(e) Reproductions (f) Re-grasp behavior (g) Without stabilizing policy (h) Without goal attractor policy

Fig. 5. Demonstrations and reproductions of the force-conditioned grasping experiment: (a) illustrates the demonstrated grasp and goal pose, (b) shows the reproduced trajectories from the quantitative evaluation (upper part of Table 2) in the x-z-plane, (c) illustrates the demonstrated and reproduced task-parameters from the quantitative evaluation, (d) presents the reproduced task-parameter evolution for a trial with multiple re-grasps where the gripper is manually opened by the user, (e), (f) display the pre- and post-grasp MoE vector fields at s_0 and s_I for this trial and (g), (h) display the vector fields and observed robot trajectories when removing either the stabilizing or the goal attractor policy within the context-adaptive policy framework.

Table 2
Quantitative evaluation (average success S, iterations I, distance D, collisions *Coll.*) on the real-robot experiments described in Sections 5.3 and 5.4. The results are reported as the mean and standard deviation computed over 20 trials.

Exp.	Method	S [%]	I	D	*Coll.*
Section 5.3	GPR (LfD policy 4.1, [2])	0.0 ± 0.0	1000.0 ± 0.0	0.096 ± 0.037	16
	GPR & stab. (4.2)	55.0 ± 49.7	694.1 ± 357.5	0.022 ± 0.004	**0**
	GPR & goal attr. (4.3)	5.0 ± 21.8	851.5 ± 257.2	0.088 ± 0.043	16
	Proposed framework	$\mathbf{100.0 \pm 0.0}$	$\mathbf{264.3 \pm 68.1}$	$\mathbf{0.017 \pm 0.006}$	**0**
	Diffusion Policy ([20])	0.0 ± 0.0	–	0.076 ± 0.030	20
Section 5.4	GPR (LfD policy 4.1, [2])	0.0 ± 0.0	1000.0 ± 0.0	0.148 ± 0.038	7
	GPR & stab. (4.2)	0.0 ± 0.0	1000.0 ± 0.0	0.024 ± 0.002	**0**
	GPR & goal attr. (4.3)	20.0 ± 40.0	839.3 ± 291.3	0.097 ± 0.074	3
	Proposed framework	$\mathbf{95.0 \pm 21.8}$	$\mathbf{370.1 \pm 165.0}$	$\mathbf{0.019 \pm 0.004}$	**0**
	Diffusion Policy ([20])	0.0 ± 0.0	1000.0 ± 0.0	0.211 ± 0.226	19

5.4. Manipulation of deformable food items

In the third experiment, we demonstrate our approach's adaptation to different placing strategies of a deformable object, in this case a fish fillet. Using a similar robot and gripper as in Section 5.3, but with different fingertips in order to facilitate the grasp, we record the placing of the fish on a tray via kinesthetic teaching, as displayed in Fig. 1. Depending on the grasp configuration, the placing is performed differently, by approaching the tray from the side where the fillet hangs. We provide four demonstrations for each side. To capture the in-gripper fish geometry, we fine-tune YOLOv7 [36] on the described use-case to obtain two bounding boxes, one for each side of the gripper. We define the task-parameters as the fish's overlap on each side of the gripper, i.e. $\boldsymbol{c} = \begin{bmatrix} d_{\text{left}} & d_{\text{right}} \end{bmatrix}^\top$, where $d_i = \max_k \left\| \begin{bmatrix} u_{k,i} & v_{k,i} \end{bmatrix}^\top - \begin{bmatrix} u_{ee} & v_{ee} \end{bmatrix}^\top \right\|$ is the maximum distance between the end-effector pose $\begin{bmatrix} u_{ee} & v_{ee} \end{bmatrix}^\top$, transformed to image coordinates, and the $k \in [0, 4]$ corners of the respective bounding box $\begin{bmatrix} u_{k,i} & v_{k,i} \end{bmatrix}^\top$.[15] Fig. 6(a) clarifies the definition of $\boldsymbol{c}$ and Fig. 6(c) shows the demonstrated task-parameters together with the reproduced ones. Including robot position and orientation, the input dimension is $I = 8$.

We perform an identical quantitative evaluation as in Section 5.3, where the context-adaptive policy is used to compute the end-effector velocity $\{\dot{\boldsymbol{x}}_d, \dot{\boldsymbol{\phi}}_d\}$ in a 20 Hz loop, hence $\mathcal{O} = 6$. The results are shown in Table 2 and the reproduced task-parameters are illustrated in Fig. 6(c). Moreover, Fig. 6(e) displays reproduced trajectories for both fish fillet configurations when starting at the demonstrated start poses and Fig. 6(b), 6(d) and 6(f) exemplify the vector fields and the reproduced trajectory of a single trial at different times in case the fish fillet hangs on the right.

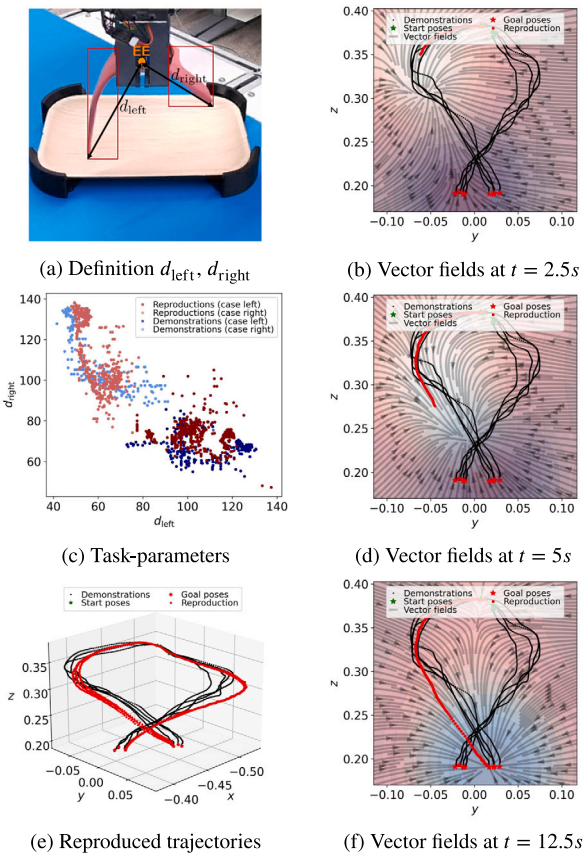
Table 3
Ablation analysis of the stabilizing policy's orientation extension on the experimental setup described in Section 5.3. The rows denote the equation number used for the stabilizing policy: (14) corresponds to the ablated stabilizing policy without the orientation component, and (8) corresponds to our framework with the orientation component. The evaluated metrics (average success S, iterations I, distance D, collisions *Coll.*) are reported as the mean and standard deviation computed over 20 trials.

Method	S [%]	I	D	*Coll.*
Eq. (14)	5.0 ± 21.8	969.5 ± 125.3	0.42 ± 0.22	1
Eq. (8)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{320.1 \pm 29.6}$	$\mathbf{0.03 \pm 0.02}$	**0**

5.5. Ablation analysis of the stabilizing policy's orientation extension

The previous sections aimed to evaluate the effect of the proposed framework compared to other non-MoE baselines and different combinations of the individual policies. To isolate the contribution of the stabilizing policy's orientation component (Contribution 3, see Section 1), we perform an ablation where we remove it from Eq. (8).

[15] As this work focuses on the context-adaptive policy framework, the deformable object detection is implemented in this simplified, but sufficient form. However, more complex or 3D-based detectors would also be conceivable if the task-parameter dimension remains treatable by GPR (see 6).



(a) Definition d_{left}, d_{right} (b) Vector fields at $t = 2.5s$

(c) Task-parameters (d) Vector fields at $t = 5s$

(e) Reproduced trajectories (f) Vector fields at $t = 12.5s$

Fig. 6. Demonstrations and reproductions of the manipulation of deformable food items experiment: (a) clarifies the definition of the task-parameters, (c) displays the reproduced trajectories of each four trials for both fish fillet configurations when starting at the demonstrated start poses, (e) shows the demonstrated and reproduced task-parameters from the quantitative evaluation (lower part of Table 2) and (b), (d), (f) exemplify the vector fields and the reproduced trajectory of a single trial at $t = 2.5s$, $t = 5s$ and $t = 12.5s$ respectively when the fish fillet hangs on the right.

Specifically, we discard the orientation term and compute the gradient with respect to the current position $\boldsymbol{x}_*$ using only $v_{c,\boldsymbol{x}}$, i.e.,

$$\boldsymbol{\mu}_{\text{sp}} = -K_{\text{sp},\boldsymbol{x}} \frac{\nabla_{\boldsymbol{x}_*} v_{c,\boldsymbol{x}}}{\max\left(\|\nabla_{\boldsymbol{x}_*} v_{c,\boldsymbol{x}}\|, \epsilon\right)} v_{c,\boldsymbol{x}}. \tag{14}$$

For this ablation, we use the experimental setup of Section 5.3, but sample the starting poses from an extended range to explicitly examine the effects of the stabilizing policy's orientation component, which has an impact only in out-of-distribution orientations. We sample the starting poses around the demonstrated goal poses by perturbing translations uniformly within a sphere of radius 0.15 m and rotations uniformly about arbitrary axes with a maximum angle of 45°. The results are reported in Table 3. The rows in this table denote the stabilizing policy used: Eq. (14) corresponds to the ablated policy, which omits the orientation component, while Eq. (8) corresponds to our full framework, which includes it. The behavioral differences arising from this ablation are illustrated in Fig. 7, where the upper and lower rows correspond to the ablated and full policy, respectively, consistent with Table 3. As shown, our full framework recovers from out-of-distribution orientations, whereas the ablated policy fails to do so. The same execution sequence is presented in the accompanying video.

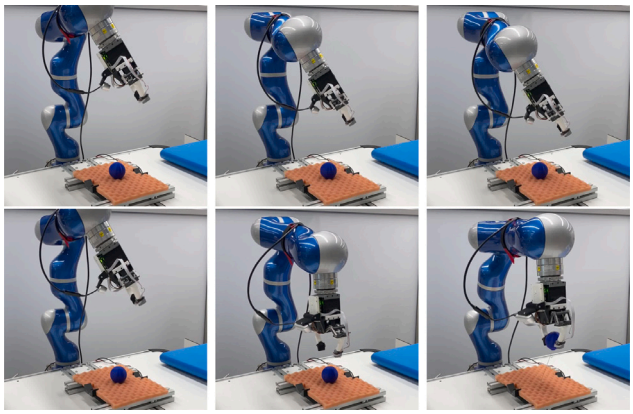


Fig. 7. Behavioral differences between the ablated and full stabilizing policy across a representative execution sequence. The figure presents a time progression (from left to right) of the reproduction, with three snapshots captured at sequential time steps. The upper and lower rows correspond to the ablated stabilizing policy (Eq. (14)) and the full framework (Eq. (8)), respectively. Quantitative performance metrics for this sequence are reported in Table 3.

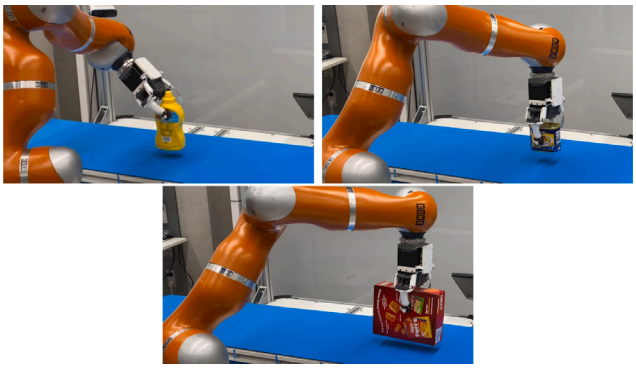


Fig. 8. *Placing* phase of the object-centric grasping experiment: The robot places the grasped objects ("Mustard Bottle", "Potted Meat Can" and "Cracker Box") outside the workspace on the conveyor belt.

5.6. System extension — Object-centric grasping

In the final experiment, we apply our approach to grasp three selected YCBV objects[16] from a conveyor belt. We record three demonstrations per object via kinesthetic teaching with static conveyor belt consisting of a *grasping*, *placing* and *resting* phase. During the *grasping* phase the object is approached and grasped within a pre-defined region of the workspace, before being placed again on the conveyor belt outside the pre-defined region during the *placing* phase. After placing the object, the robot moves to a defined position in which it waits (*resting* phase) until the object moves back into the pre-defined region. The *Placing* phase, is exemplarily illustrated in Fig. 8 for each object.

To robustly accomplish the task, three extensions to the previously introduced framework were made. First, the continuous demonstrations were divided by phase into three datasets. Two separate policies were trained: one on the *grasping* phase dataset transformed to object coordinate system, and another on the remaining two datasets in world coordinate system. Depending on the current phase, the respective policy is selected, allowing the object to be grasped at any location within the workspace while the *placing* and *resting* poses remain static. Moreover, a non-zero mean is used for the gripper joint velocity within the *grasping* phase so that the gripper re-opens while approaching the

[16] Available at https://rse-lab.cs.washington.edu/projects/posecnn/.

Table 4
Quantitative evaluation (average success S, iterations I, distance D, collisions *Coll.*) on the experiment described in 5.6.

Exp.		Obj.	S [%]	I	D	*Coll.*
Object pose variation		Cracker Box	100.0 ± 0.0	387.4 ± 75.6	0.059 ± 0.015	0
		Mustard Bottle	94.4 ± 22.9	376.8 ± 70.8	0.159 ± 0.077	1
		Potted Meat Can	100.0 ± 0.0	386.3 ± 98.2	0.113 ± 0.069	0
Robot pose variation		Cracker Box	100.0 ± 0.0	309.9 ± 54.8	0.077 ± 0.023	0
		Mustard Bottle	93.3 ± 24.9	348.2 ± 33.7	0.101 ± 0.014	1
		Potted Meat Can	93.3 ± 24.9	311.3 ± 32.5	0.121 ± 0.076	1
Dynamic grasping	Speed: $0.01 \frac{m}{s}$	Cracker Box	100.0 ± 0.0	392.7 ± 103.3	0.072 ± 0.020	0
		Mustard Bottle	83.3 ± 37.3	353.0 ± 63.6	0.220 ± 0.162	1
		Potted Meat Can	100.0 ± 0.0	333.2 ± 63.3	0.108 ± 0.041	0
	Speed: $0.02 \frac{m}{s}$	Cracker Box	83.3 ± 37.3	461.4 ± 163.8	0.093 ± 0.091	1
		Mustard Bottle	83.3 ± 37.3	387.4 ± 134.4	0.202 ± 0.128	1
		Potted Meat Can	100.0 ± 0.0	422.0 ± 84.6	0.089 ± 0.022	0
	Speed: $0.03 \frac{m}{s}$	Cracker Box	100.0 ± 0.0	401.2 ± 121.2	0.062 ± 0.018	0
		Mustard Bottle	50.0 ± 50.0	397.0 ± 40.0	0.249 ± 0.150	3
		Potted Meat Can	66.7 ± 47.1	559.5 ± 44.4	0.138 ± 0.058	2

object. Finally, a Matérn kernel [2] is used for $k(\boldsymbol{x}_*, \boldsymbol{x}_g)$ and $k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_g)$ in Eq. (9), producing higher end-effector velocities near the goal poses and thus allowing higher object velocities.

The object position is tracked at 20 Hz using a combination of learning and model-based computer vision methods, including YOLOv7 [36], a 6D object pose estimation [37] and a 3D Object Tracking [38]. We define the task-parameters as $\boldsymbol{c} = \left[k_{\text{obj}}\ j_{\text{phase}}\right]^\top$, where $k_{\text{obj}} \in \{2, 5, 9\}$ represents "Cracker Box", "Mustard Bottle", and "Potted Meat Can". Further, $j_{\text{phase}} \in \{0, 1, 2\}$ represents the *grasping*, *placing* and *resting* phase, that are active when the object is located within the pre-defined region, grasped or located outside the pre-defined region, respectively. Thus, the input and output dimensions are $\mathcal{I} = 8$ and $\mathcal{O} = 7$.

We demonstrate our extended framework's capability to grasp the three objects on both static and moving conveyor belt, with varying initial objects and robot poses. The experimental setup and evaluated metrics match those of the previous experiments. The results of this experiment are presented in Table 4, split into three sub-experiments:

- Object pose variation: We evaluate our approach's generalization to unseen object configurations by varying both object position and orientation while keeping the robot's starting pose fixed at the resting configuration. The object is placed at the six positions defined by the intersections of $x \in \{-0.15, 0.0, 0.15\}$ m and $y \in \{0.45, 0.60\}$ m, combined with three orientations around the z-axis relative to the demonstrated configuration ($\{-45°, 0°, 45°\}$), resulting in 18 distinct object configurations per object.
- Robot pose variation: We assess the robustness of our approach to varying initial robot configurations while the object remains static at a single pose ($[x, y] = [0.15, 0.525]$ m with the demonstrated orientation). The robot starting pose is uniformly sampled 15 times per object from the range of demonstrated poses.
- Dynamic grasping: We test our approach's capability to grasp objects on a moving conveyor belt under varying motion conditions. The conveyor operates at three different speeds (0.01, 0.02, and 0.03 m/s), with the object placed at two positions ($y \in \{0.45, 0.60\}$ m) and three orientations around the z-axis relative to the demonstrated configuration ($\{-45°, 0°, 45°\}$), yielding 18 dynamic test conditions per object.

For visual reference, refer to the accompanying video, where we also showcase how this extension can be used in the context of human-robot handovers (screenshots in Fig. 9).

6. Discussion

In this section, we discuss the results of the conducted experiments, analyze failure cases, and outline the limitations of our approach.

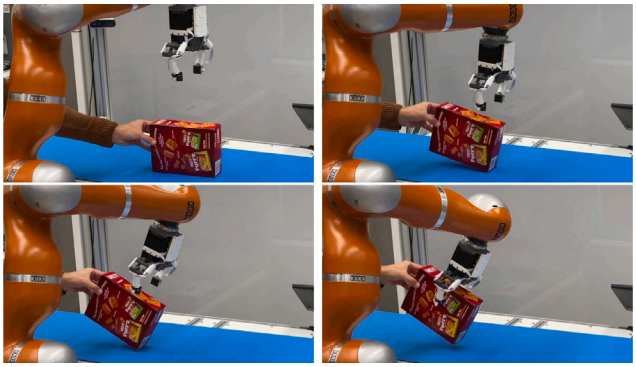


Fig. 9. Possible extensions of our approach include online adaptation of robot behavior to objects poses, e.g., during handovers: The robot tracks the dynamically changing position of the object ("Cracker Box") in the human hand, continuously adjusting its trajectory in real time to successfully complete the grasp.

6.1. Baseline performance

The low success rates (S) of both GPR (standalone LfD policy, Section 4.1) and the considered Diffusion Policy (DP) baseline (Tables 1 and 2) underscore the challenge of learning reliable and robust behavior from limited demonstrations. The results demonstrate that these policies, when deployed without additional tuning or architectural adaptations, often lack the robustness required in small-data regimes.[17] In our experiments, the observed low success rates can be primarily attributed to either (1) insufficient state-space coverage, due to the small number of demonstrations provided,[18] or (2) a lack of empirical convergence within the tested horizon.

The insufficient state-space coverage causes both baselines, to frequently operate outside the support of the training distribution when exposed to novel initial states or deviations from the demonstrated trajectory. This often results in unpredictable, potentially safety-critical, behavior, as reflected in the metrics capturing safety and trajectory fidelity, namely the number of collisions (C) and the distance to demonstrated trajectories (D). Furthermore, as visible in the accompanying

[17] This observation specifically pertains to the considered, unmodified baseline configurations with limited data, directly motivating the structural enhancements introduced in our approach.

[18] Note that a small-data regime was deliberately chosen to assess the robustness of the proposed method under limited data availability. However, the large parameter count in DP contrasts sharply with the limited number of demonstrations, making effective training particularly challenging.

video, both baselines fail to halt upon reaching the goal state, instead continuing to drift slowly without achieving zero velocity. This lack of empirical convergence within the tested horizon is reflected in the number of iterations (I) and the timeouts (reaching the iteration limit) indirectly depicted therein. Both of these failure cases are highly evident in the real-world experiments (see Table 2), where DP resulted in 39 collisions and 1 timeout, while GPR caused 23 collisions and 17 timeouts, across 40 trials per policy. Since a zero-mean GP is used, causing GPR to predict near-zero velocities in regions far from the training data, a higher number of timeouts and a lower number of collisions are observed for GPR compared to DP.

6.2. Proposed framework performance

In contrast, the combination of multiple *expert* policies informed by the epistemic uncertainty (proposed framework, Section 4) is able to achieve high success and low collision rates while maintaining proximity to the demonstrated data distribution. Accordingly, Table 1 shows that, in scenarios without external task-parameter inputs, our approach performs comparably to the established DS-based method SEDS in terms of success rate,[19] but superior in terms of adhering to demonstrations. However, as discussed in Sections 2 and 4, SEDS does not support modulation by external task parameters, a capability offered by our method, making a direct comparison against SEDS on the real-robot experiments not possible. We successfully show this modulation capability in three real-robot experiments (Sections 5.3, 5.4 and 5.6), where the context-adaptive policy dynamically adjusts its behavior in response to variations in both continuous and discrete task-dependent parameters during execution. The results in Table 2, particularly the combined success rate of 97.5% (the few failures are due to reaching the iteration limit) and 0 collisions, suggest that our approach can be robustly deployed on real robots, providing important properties like intuitive and data-efficient skill transfer, as well as improved adaptability to changing environments.

The experimental results also show that both supplementary policies, naturally emerging from the properties of the Gaussian Process framework with RBF kernels (as in Section 4), fulfill their intended purpose in the considered scenarios and that both are required to achieve robust reproduction behaviors, as presented quantitatively in Tables 1 and 2 and visually in Figs. 5(g) and 5(h). Thus, removing the stabilizing or the goal attractor policy either causes the robot to drift away from the demonstrations or reduced empirical convergence within the tested horizon, respectively. Complementing these findings, the ablation analysis, reported in Table 3, demonstrates the benefit of incorporating orientation into the stabilizing policy. The results show that the proposed formulation recovers from out-of-distribution orientations without the need for an additional recovery mechanism, whereas removing the orientation component prevents such recovery.

Figs. 5(b) and 6(c) further demonstrate that our approach robustly performs the tasks even when the task-dependent parameters encountered during reproduction (red) differ, in some cases substantially, from those observed in the demonstrations (blue). In addition, we show in Table 4 that our task-space formulation allows the context-adaptive policy to be represented locally with respect to the objects, which, in combination with the real-time reactivity of our approach, enables the robot to dynamically adjust its behavior to continuously changing object poses while maintaining robustness. This reactivity is demonstrated in all experiments and is a direct benefit of the DS-based formulation. Moreover, we exemplify that our formulation yields hyperparameters applicable across multiple skills within the same domain, reducing computational effort.

[19] The few failures of SEDS are due to reaching the iteration limit.

6.3. Failure cases and limitations

Although our method demonstrates robust performance in both simulation and real-robot evaluations, we identify a few possible failure regimes.

False attractors generated by the LfD policy near the reference trajectory can persist in regions where $\pi_{\text{lfd}} \approx 1$, causing the robot to stall. The emergence of these false attractors hinges on the extrapolation capabilities of the LfD policy, yet, in our framework, this limitation can be mitigated by choosing a small kernel length. This results in a widespread activation area of the stabilizing policy, suppressing the false attractors. However, it introduces a trade-off, as shorter kernel lengths enforce more restrictive behavior, thereby limiting the system's extrapolation range. Moreover, spurious equilibria can emerge in complex scenarios when competing policies produce matching velocity magnitudes in opposing directions, similarly resulting in stalled motion. In addition, the variance gradient can saturate far from the training distribution, making recovery with the proposed stabilizing policy formulation infeasible for such states. Increasing the kernel length expands the recovery region, but simultaneously broadens the LfD activation area, which may introduce the false attractors described above. Nonetheless, the aforementioned failure modes are largely resolvable through the proposed hyperparameter optimization, evidenced by the 100% success rates achieved in our simulated experiments (Section 5.2), in which the hyperparameters were optimized as described in Section 5.1.

Further, initial starting positions near the goal may cause the robot to shortcut directly to the target without replicating the expert trajectory. Incorporating task-completion states (e.g., phase indicators) into task-dependent parameters, as demonstrated in Section 5.6, effectively mitigates the issue.

Additionally, the argmax selection creates Voronoi boundaries, at which the policy switches abruptly between goal points, creating discontinuities in the vector field. In practice, however, the underlying Cartesian impedance controller mitigates these discontinuities by inherently imposing velocity limits and filtering rapid command transitions. A common strategy employed in SEDS-like approaches to address this limitation is to transform all demonstration data into a goal-centric reference frame, ensuring that all demonstrations share the same attractor point at the origin. While this could in principle be applied to our method, it assumes that a single goal-centered dynamical system can explain all demonstrations — a premise that does not hold in all our experimental settings (e.g., Sections 5.4 and 5.6).

In the real-robot experiments that incorporate continuous task-dependent parameters (Sections 5.3 and 5.4), failures predominantly arise from OOD $\boldsymbol{c}$ values where the LfD policy provides inadequate guidance. Under these conditions, the stabilizing policy merely constrains the robot to the vicinity of the demonstrations, without contributing to task completion, often causing the robot to stall or oscillate within a limited range. Addressing this limitation requires a stabilizing policy that not only steers the robot back toward the training distribution but also actively manipulates the environment to bring back task-relevant parameters into the known regime — an extension we leave for future work.

Furthermore, self-collisions and collisions with the environment and target objects remain practical constraints, as evidenced in Table 4. While self-collisions occur only rarely and can be prevented through joint limit checks, the majority of failure cases stem from unintended collisions with target objects resulting from misalignment between the object and robot during the grasp attempt. These collisions typically trigger a safety stop, but may also result in objects being displaced or knocked over. In such cases, the altered relative grasping trajectories can intersect with environmental obstacles, leading to collisions with the environment. Further, collisions with objects increase with belt velocity due to an increased misalignment between the object and gripper during the grasp attempt. This behavior stems from the reactive

nature of our learned policy, which responds to changes but lacks feedforward or anticipatory planning capabilities. Pose measurement errors also contribute to these failures, though their individual impact remains unquantified. We plan to mitigate these issues by integrating an explicit obstacle-avoidance policy into the MoE framework.

In addition, the Euler-vector RBF kernel does not strictly respect SO(3) topology, resulting in wrap-around effects near $\pm\pi$, introducing discontinuities. In practice, this limitation was mitigated in our experiments by expressing all orientations relative to the first demonstrated configuration, as empirically shown in real-robot experiment results, where we did not observe this behavior. To address this limitation, manifold-aware kernels will receive attention in future work.

Moreover, current limitations include the individualized representation of task-parameters, the limited scalability arising from the kernel-based formulation, and the GPR-induced restricted input vector dimensionality. Accordingly, the proposed method is specifically designed for and most effective in low-dimensional, small-data scenarios — a common setting in many real-world tasks. For high-dimensional sensor inputs, by contrast, feature extraction needs to be performed decoupled from the policy, as implemented in Sections 5.4 and 5.6 or in a more generic way to enable zero-shot transfer. However, the modularity of our formulation not only permits the extension of the framework with application-dependent policies, but also, in principle, the incorporation of alternative LfD policy representations, subject to their ability to provide uncertainty quantification.

7. Conclusion

We introduced a DS-based Mixture-of-Experts framework that can be modulated by generic low-dimensional task-parameters, making it well-suited for reactive object manipulation including soft, deformable objects. We showed the applicability of our solution in both simulation and three manipulation tasks on a real 7-DoF robotic system. In future work, we will investigate the use of alternative LfD policy representations, such as diffusion or flow-based policies, that can handle more diverse inputs while simultaneously leveraging the robustness provided by our framework.

CRediT authorship contribution statement

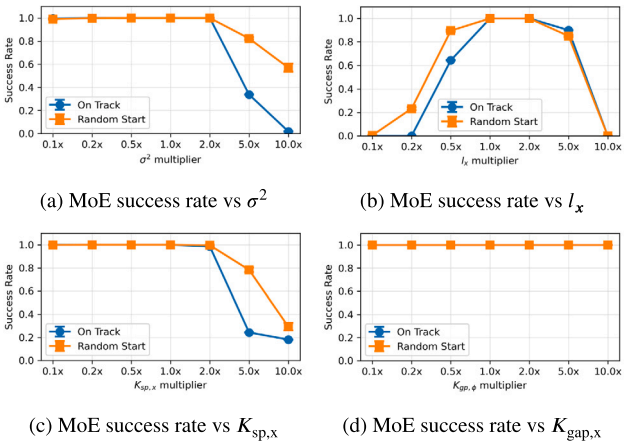
Tim R. Winter: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Leonard Klüpfel:** Software, Data curation. **Ashok M. Sundaram:** Writing – review & editing, Resources, Project administration. **Werner Friedl:** Resources. **Maximo A. Roa:** Resources, Project administration, Funding acquisition. **Freek Stulp:** Writing – review & editing, Resources, Funding acquisition. **João Silvério:** Writing – review & editing, Supervision, Resources, Methodology, Conceptualization.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the authors used Qwen3.6 and similar models in order to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Funding

This work has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No. 101070600, project SoftEnable.



(a) MoE success rate vs σ^2 (b) MoE success rate vs l_x

(c) MoE success rate vs $K_{\text{sp,x}}$ (d) MoE success rate vs $K_{\text{gap,x}}$

Fig. A.1. Hyperparameter sensitivity analysis: Success rate of our MoE approach measured with different multipliers applied to the CMA-ES optimized base value for (a) σ^2, (b) l_x, (c) $K_{\text{sp},x}$, and (d) $K_{\text{gap},x}$. Results are shown for on-demonstration (blue) and random starting poses (orange), with error bars indicating the standard deviation across 50 trials. The multiplier 1.0 corresponds to the optimized value.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Hyperparameters

Table A.1 summarizes all experiment-dependent parameters and hyperparameters used throughout the experiments described in Sections 5.2 to 5.4 and 5.6, for both the full framework and combinations of individual policies. All hyperparameters (columns σ^2 - $K_{\text{gap},\phi}$) are obtained via CMA-ES optimization as described in Section 5.1. Vector-valued hyperparameters are split into individual columns, and missing values are indicated by "–".

Table A.2 lists the hyperparameters used for the Diffusion Policy across all experiments. These values were the same for both the LASA and real-robot tasks. The notation for the hyperparameters is adopted from [20].

A sensitivity analysis on the LASA handwriting dataset (Section 5.2), which shows how variations in the hyperparameters affect the success rate of the proposed framework, is displayed in Fig. A.1, demonstrating robust performance across a reasonable range of hyperparameter values around the optimized solution.

Appendix B. Runtime performance

To evaluate the practical viability of the proposed framework for reactive manipulation, we report inference times, hardware specifications, and a scaling analysis with respect to the number of demonstrations N.

We conducted an inference time comparison on the experimental setup described in Section 5.2, where we measure the inference time over 200 trials with multiple inferences per trial. The results are presented in Table B.3.

Simulation evaluations were conducted on a workstation equipped with a 13th Gen Intel Core i5-13500 CPU (20 logical cores), 16 GB RAM, and an NVIDIA GeForce GTX 1080 Ti GPU (11 GB VRAM). Our method is implemented in standard Python without speed optimizations, parallelization, or GPU acceleration. For direct comparison we also report inference times for SEDS and Diffusion Policy, with SEDS

Table A.1
Parameters and hyperparameters used in each experiment. Vector-valued hyperparameters are split into individual columns. Values apply to both the full framework and combinations of individual policies.

Exp.	$\mathcal{I}$	$\mathcal{O}$	N	$I_{\max}$	Seed	σ^2	l_c	l_x	l_ϕ	$K_{\text{sp},x}$	$K_{\text{sp},\phi}$	$K_{\text{gap},x}$	$K_{\text{gap},\phi}$
Section 5.2	2	2	500	500	0	1.471	–	3.800	–	49.955	–	84.870	–
Section 5.3	8	7	500	1000	42	0.432	8.095	0.066	0.074	0.280	0.627	0.281	0.782
Section 5.4	8	6	500	1000	42	0.432	20.095	0.066	0.074	0.280	0.627	0.281	0.782
Section 5.6	8	7	500	1000	42	0.432	0.095	0.066	0.074	0.280	0.627	0.281	0.782

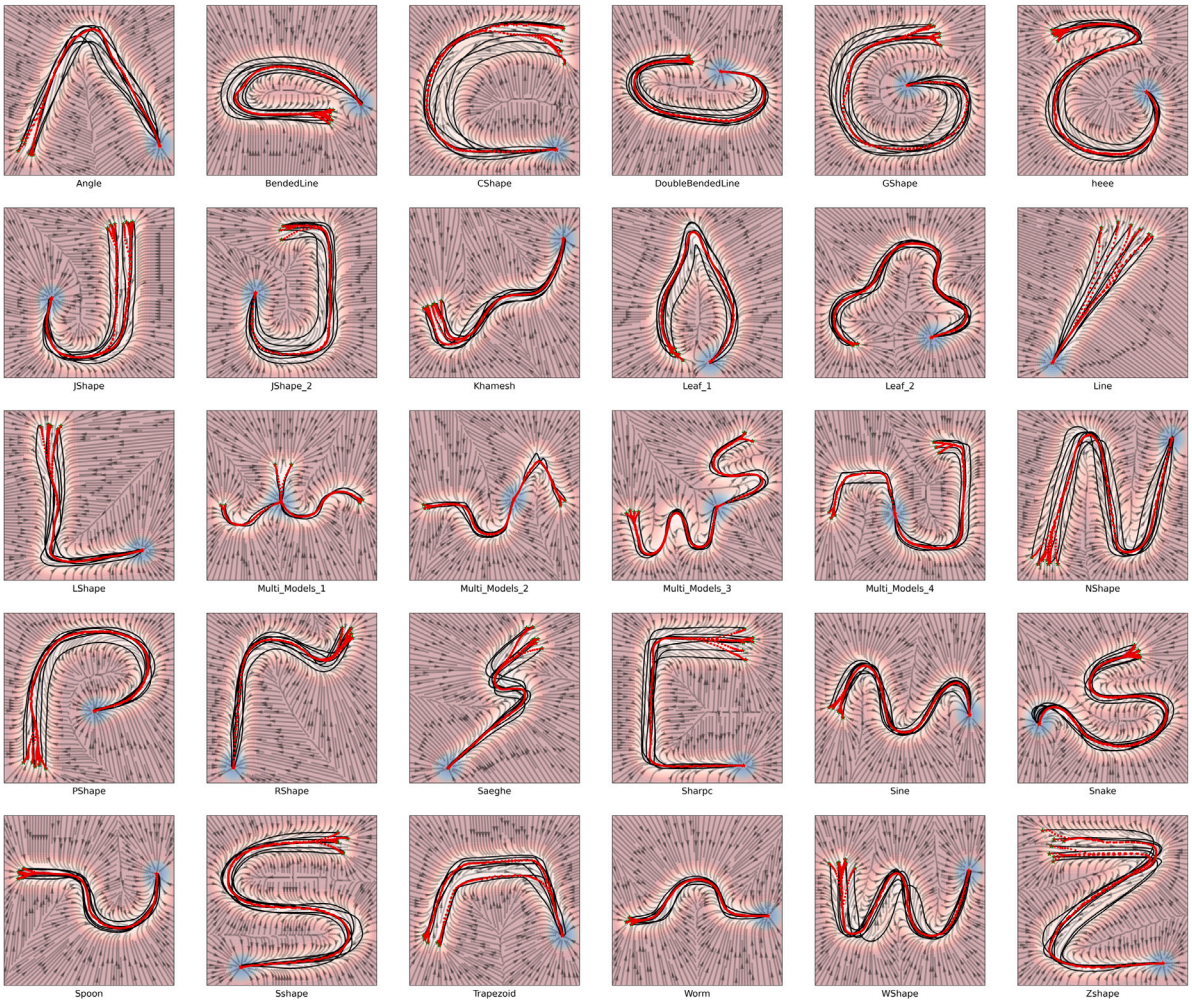


Fig. A.2. Reproductions for all letters of the LASA handwriting dataset when $\boldsymbol{x}_{0,k}$ is placed at the initial pose of each of the seven demonstrations showing the executed paths, vector fields and mixing coefficients of the MoE (refer to Fig. 3 for clarification).

being trained using the provided MATLAB code, and inference being executed using a standard Python GMR implementation,[20] where the GMM is initialized with the trained parameters, running entirely on CPU. Diffusion Policy was evaluated using the state-based environment,[21] which leverages GPU acceleration. The table confirms that our framework is computationally lightweight and does not require specialized hardware.

Real-robot experiments were conducted on a separate workstation with an Intel Core i9-14900K CPU (32 logical cores), 64 GB RAM, and an NVIDIA RTX 4500 Ada Generation (24 GB VRAM). While formal runtime benchmarking was not conducted on the physical platform, our implementation, despite the unoptimized implementation, comfortably satisfies the real-time control frequencies required for reactive manipulation.

We further investigate how our approach scales with the number of demonstration samples N using the experimental setup described in Section 5.2. All experiments employ the hyperparameters optimized for $N = 500$ (Table A.1). As shown in Fig. B.3(a), the inference time grows quadratically with N, consistent with the $O(N^2)$ complexity of

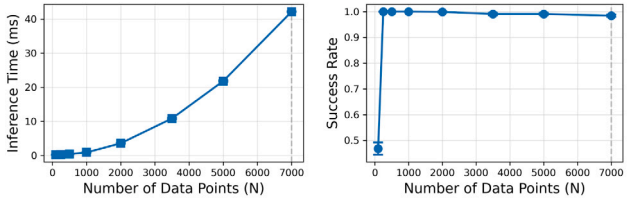
[20] Available at https://github.com/AlexanderFabisch/gmr.
[21] Adapted from https://github.com/real-stanford/diffusion_policy to match our dataset and environment.

Table A.2
Diffusion Policy hyperparameters used across all experiments.

Hyperparameter	Value
Observation horizon, T_o	2
Prediction horizon, T_p	16
Action horizon, T_a	8
Diffusion steps (train), D-Iters Train	100
Diffusion steps (eval), D-Iters Eval	10
Number of training epochs	100
Total number of NN parameters	~ 65.3 M

Table B.3
Inference time comparison on the experimental setup described in Section 5.2.

Method	Inf time [ms]
Our approach (MoE)	0.402 ± 0.002
SEDS [3]	0.373 ± 0.001
Diffusion Policy [20]	1035.64 ± 16.33



(a) Inference time over N (b) Success rate over N

Fig. B.3. Scaling behavior with respect to the number of demonstration samples N: (a) shows that the inference time per query increases quadratically with N, aligning with the theoretical $O(N^2)$ cost of Gaussian process inference. (b) demonstrates that the success rate rapidly saturates, forming a plateau for $N \geq 250$. Vertical bars denote ± 1 standard deviation across five seeds, each consisting of 10 trials per letter. The vertical dashed line marks the full LASA handwriting dataset size ($N = 7000$).

Gaussian process predictive evaluation. In contrast, the success rate quickly saturates, reaching a plateau for $N \geq 250$ (Fig. B.3(b)). This behavior indicates that a modest number of demonstrations is sufficient to adequately cover the underlying task distribution. Consequently, our chosen configuration $N = 500$ resides comfortably on the performance plateau while maintaining computationally efficient inference times, and the outer control loop frequency is set with a buffer large enough to ensure that it is met even in the event of significant fluctuations of the inference times.

Appendix C. Empirical robustness to c-perturbations

To characterize the practical robustness and tolerance bounds with respect to c-perturbations, we conduct an ablation study on the experimental setup described in Section 5.3. In this analysis, c is artificially perturbed by introducing constant biases Δc of varying sign during reproduction to simulate systematic force offsets or drift. As shown in Fig. C.4, the proposed framework achieves full success rates under perturbations up to $\pm 2.5\,\mathrm{N}$, demonstrating effective tolerance to realistic c-fluctuations. We note that the tolerable deviation envelope depends strongly on the task-parameter kernel length l_c (set to $l_c \approx 8$, as detailed in Table A.1).

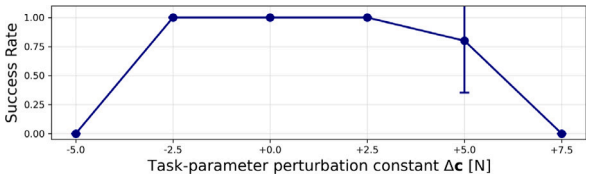


Fig. C.4. Empirical robustness to c perturbations. Constant biases Δc (in N) are added to c during reproduction to simulate systematic force or tracking offsets. The y-axis reports task success rate (Section 5.3); error bars represent ± 1 standard deviation over trials. With $l_c \approx 8$ (Table A.1), the system maintains full success for deviations up to $\pm 2.5\,\mathrm{N}$, confirming practical tolerance to realistic c fluctuations.

Appendix D. Supplementary data

A video showing all experiments, including the LASA benchmarks (Section 5.2) and real-robot tasks (Sections 5.3–5.6), is provided as supplementary material. It includes both demonstrations and executions, illustrating the performance of the proposed framework.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2026.105649.

Data availability

Data will be made available on request.

References

[1] H. Ravichandar, A.S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Annu. Rev. Control. Robot. Auton. Syst. 3 (1) (2020) 297–330, http://dx.doi.org/10.1146/annurev-control-100819-063206.

[2] C.E. Rasmussen, Gaussian Processes for Machine Learning, MIT Press, 2006, p. 272.

[3] S.M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with Gaussian mixture models, TRO 27 (5) (2011) 943–957, http://dx.doi.org/10.1109/tro.2011.2159412.

[4] E. Pignat, S. Calinon, Bayesian Gaussian mixture model for robotic policy imitation, RA-L 4 (4) (2019) 4452–4458, http://dx.doi.org/10.1109/lra.2019.2932610.

[5] G. Franzese, A. Mészáros, L. Peternel, J. Kober, ILoSA: Interactive learning of stiffness and attractors, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, IROS, 2021, pp. 7778–7785, http://dx.doi.org/10.1109/IROS51168.2021.9636710.

[6] A. Meszaros, G. Franzese, J. Kober, Learning to pick at non-zero-velocity from interactive demonstrations, RA-L 7 (3) (2022) 6052–6059, http://dx.doi.org/10.1109/lra.2022.3165531.

[7] S.M. Khansari-Zadeh, SEDS & LASA Handwriting Dataset. URL https://cs.stanford.edu/people/khansari/download.html.

[8] S. Calinon, F. Guenter, A. Billard, On learning, representing, and generalizing a task in a humanoid robot, IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 37 (2) (2007) 286–298, http://dx.doi.org/10.1109/tsmcb.2006.886952.

[9] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intell. Serv. Robot. 9 (1) (2015) 1–29, http://dx.doi.org/10.1007/s11370-015-0187-9.

[10] D. Nguyen-Tuong, J. Peters, Local Gaussian process regression for real-time model-based robot control, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, IROS, IEEE, 2008, pp. 380–385, http://dx.doi.org/10.1109/iros.2008.4650850.

[11] C. Atkeson, Using locally weighted regression for robot learning, in: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), vol. 2, 1991, pp. 958–963, http://dx.doi.org/10.1109/ROBOT.1991.131713.

[12] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: Learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373, http://dx.doi.org/10.1162/neco_a_00393.

[13] A. Paraschos, C. Daniel, J.R. Peters, G. Neumann, Probabilistic movement primitives, in: Advances in Neural Information Processing Systems (NeurIPS), vol. 26, Curran Associates, Inc., 2013, URL https://proceedings.neurips.cc/paper_files/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

[14] Y. Huang, L. Rozo, J. Silvério, D.G. Caldwell, Kernelized movement primitives, IJRR 38 (7) (2019) 833–852, http://dx.doi.org/10.1177/0278364919846363.
[15] J. Silverio, L. Rozo, S. Calinon, D.G. Caldwell, Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, IROS, IEEE, 2015, http://dx.doi.org/10.1109/iros.2015.7353413.
[16] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, O. Sigaud, Learning compact parameterized skills with a single regression, in: Proc. IEEE Intl Conf. on Humanoid Robots, Humanoids, IEEE, 2013, http://dx.doi.org/10.1109/humanoids.2013.7030008.
[17] A. Pervez, D. Lee, Learning task-parameterized dynamic movement primitives using mixture of GMMs, Intell. Serv. Robot. 11 (1) (2017) 61–78, http://dx.doi.org/10.1007/s11370-017-0235-8.
[18] T. Kulak, H. Girgin, J.-M. Odobez, S. Calinon, Active learning of Bayesian probabilistic movement primitives, RA-L 6 (2) (2021) 2163–2170, http://dx.doi.org/10.1109/lra.2021.3060414.
[19] M. Ewerton, G. Maeda, G. Kollegger, J. Wiemeyer, J. Peters, Incremental imitation learning of context-dependent motor skills, in: Proc. IEEE Intl Conf. on Humanoid Robots, Humanoids, 2016, pp. 351–358, http://dx.doi.org/10.1109/HUMANOIDS.2016.7803300.
[20] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, S. Song, Diffusion policy: Visuomotor policy learning via action diffusion, IJRR 44 (10–11) (2024) 1684–1704, http://dx.doi.org/10.1177/02783649241273668.
[21] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, S. Liu, FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow matching for robot manipulation, AAAI 39 (14) (2025) 14754–14762, http://dx.doi.org/10.1609/aaai.v39i14.33617.
[22] J. Umlauft, S. Hirche, Learning stochastically stable Gaussian process state-space models, IFAC J. Syst. Control. 12 (2020) 100079, http://dx.doi.org/10.1016/j.ifacsc.2020.100079.
[23] Z. Jin, W. Si, A. Liu, W.-A. Zhang, L. Yu, C. Yang, Learning a flexible neural energy function with a unique minimum for globally stable and accurate demonstration learning, TRO 39 (6) (2023) 4520–4538, http://dx.doi.org/10.1109/tro.2023.3303011.
[24] R. Pérez-Dattari, J. Kober, Stable motion primitives via imitation and contrastive learning, IEEE Trans. Robot. 39 (5) (2023) 3909–3928, http://dx.doi.org/10.1109/tro.2023.3289597.
[25] G. Manek, J.Z. Kolter, Learning stable deep dynamics models, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.
[26] F. Nawaz, T. Li, N. Matni, N. Figueroa, Learning complex motion plans using neural ODEs with safety and stability guarantees, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2024, pp. 17216–17222, http://dx.doi.org/10.1109/icra57147.2024.10611584.
[27] T. Li, N. Figueroa, Task generalization with stability guarantees via elastic dynamical system motion policies, in: J. Tan, M. Toussaint, K. Darvish (Eds.), Proceedings of the 7th Conference on Robot Learning, in: Proceedings of Machine Learning Research, vol. 229, PMLR, 2023, pp. 3485–3517, URL https://proceedings.mlr.press/v229/li23b.html.
[28] S. Sun, N. Figueroa, Se(3) linear parameter varying dynamical systems for globally asymptotically stable end-effector control, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2024, pp. 5152–5159, http://dx.doi.org/10.1109/iros58592.2024.10801844.
[29] Y. Hu, F.J. Abu-Dakka, F. Chen, X. Luo, Z. Li, A. Knoll, W. Ding, Fusion dynamical systems with machine learning in imitation learning: A comprehensive overview, Inf. Fusion 108 (2024) 102379, http://dx.doi.org/10.1016/j.inffus.2024.102379.
[30] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87, http://dx.doi.org/10.1162/neco.1991.3.1.79.
[31] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods, Mach. Learn. 110 (3) (2021) 457–506, http://dx.doi.org/10.1007/s10994-021-05946-3.
[32] G. Hinton, Products of experts, in: Proc. Intl Conf. on Artificial Neural Networks (ICANN), vol. 1999, IEE, 1999, pp. 1–6, http://dx.doi.org/10.1049/cp:19991075.
[33] E. Pignat, J. Silvério, S. Calinon, Learning from demonstration using products of experts: Applications to manipulation and task prioritization, Int. J. Robot. Res. 41 (2) (2021) 163–188, http://dx.doi.org/10.1177/02783649211040561.
[34] Y. Yildirim, E. Ugur, Conditional neural expert processes for learning movement primitives from demonstration, IEEE Robot. Autom. Lett. 9 (12) (2024) 10732–10739, http://dx.doi.org/10.1109/lra.2024.3477169.
[35] N. Hansen, The CMA evolution strategy: A tutorial, 2016, http://dx.doi.org/10.48550/ARXIV.1604.00772.
[36] C.-Y. Wang, A. Bochkovskiy, H.-Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: IEEE/CVF Conf. on Computer Vision and Pattern Recognition, CVPR, IEEE, 2023, pp. 7464–7475, http://dx.doi.org/10.1109/cvpr52729.2023.00721.
[37] M. Ulmer, M. Durner, M. Sundermeyer, M. Stoiber, R. Triebel, 6D object pose estimation from approximate 3D models for orbital robotics, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 10749–10756, http://dx.doi.org/10.1109/iros55552.2023.10341511.
[38] M. Stoiber, Closing the Loop: 3D Object Tracking for Advanced Robotic Manipulation (Ph.D. thesis), Technische Universität München, 2023, p. 172, URL https://mediatum.ub.tum.de/1696259.

Tim R. Winter received the M.Sc. degree in Mechatronics and Information Technology from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2021. He is currently a Research Scientist with the Department of Cognitive Robotics, Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, Germany, where he has been working toward the Ph.D. degree since October 2023. His research focuses on data-efficient and uncertainty-aware imitation learning for contextadaptive, robust, and reactive robotic manipulation.

Leonard Klüpfel received the B.Sc. degree in industrial engineering in 2018 from the Friedrich-Alexander University of Erlangen-Nürnberg, Erlangen, Germany, and the M.Sc. degree in robotics, cognition, intelligence in 2022 from TUM, Munich, Germany. He is currently working toward the Ph.D. degree in computer science with KIT, Karlsruhe, Germany. He is a Research Scientist with the Department for Perception and Cognition, Institute of Robotics and Mechatronics, DLR, Weßling, Germany. His research focuses on computer vision for semantic scene understanding in the context of robotic manipulation with a special interest in detection and tracking of articulated objects with physical considerations and inspiration.

Ashok Meenakshi Sundaram received the M.Sc. degree in Autonomous Systems from the Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany. He is a Research Scientist with the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR), Weßling, Germany. His research focuses on robotic manipulation and grasp planning.

Werner Friedl received his Dipl.-Ing.(FH) in Mechatronic at the University of applied science In Munich and starts at DLR in 2004. 2006 he develop the torso of DLR Humanoid Justin. In the DLR Hand-Arm- project he developed the forearm of the AWIWI hand and the second version of AWIWI. Since 2015 he is responsible for the mechanical hand development at DLR. His main research focus includes variable stiffness actuation, tendon driven hands and grippers for grasping (CLASH, HCG, Smarthand, LSG).

Maximo A. Roa (Senior Member, IEEE) received the Ph.D. degree in robotics in 2009 from Universitat Politecnica de Catalunya, Catalonia, Spain, and also holds the Project Management Professional (PMP) Certification since 2016. He is a Senior Scientific Researcher at the Institute of Robotics and Mechatronics in the German Aerospace Center (DLR), heading the group of Robotic Planning and Manipulation, focused on the development and implementation of locomotion and manipulation skills at different levels for industrial, service and humanoid robots. His research interests include the development of mechatronic devices, torque-based control, and space robotics. Dr. Roa was Co-Chair of the IEEE-RAS Technical Committee on Mobile Manipulation from 2013 to 2019. He is also Member of ASME, PMI, and IAU.

Freek Stulp received the doctorate degree in computer science from the Technical University of Munich, Munich, Germany, in 2007. He is currently the Head of the Department of Cognitive Robotics, Institute of Robotics and Mechatronics, German Aerospace Center, Weßling, Germany. Previously, he was an Assistant Professor with the École Nationale Supérieure de Techniques Avancées.

João Silvério received the Ph.D. degree in robotics from the University of Genoa, Genoa, Italy, and the Italian Institute of Technology, Genoa, in 2017. He is the Group Leader of the Interactive Skill Learning Group, German Aerospace Center (DLR), Weßling, Germany, since April 2022. He was also a Postdoctoral Researcher until May 2019. From 2019 to 2022, he was a Postdoctoral Researcher with the Idiap Research Institute, Martigny, Switzerland. He is interested in machine learning for robotics, particularly imitation and reinforcement learning.