Task Transfer with Stability Guarantees via Elastic Dynamical System Motion Policies

Anonymous Author(s)
Affiliation
Address
email

Abstract: Dynamical System (DS) based Learning from Demonstration (LfD) allows learning of reactive motion policies with stability and convergence guarantees from a handful of trajectories. Yet, current DS learning techniques lack the flexibility to transfer to new task instances as they ignore explicit task parameters that inherently change the underlying trajectories. In this work, we propose Elastic-DS, a novel DS learning and transfer approach that embeds task parameters into the Gaussian Mixture Model (GMM) based Linear Parameter Varying (LPV) DS formulation. Central to our approach is the Elastic-GMM, a GMM constrained to a set of task parameters including i) geometric descriptors encoded as $SE(3)$ task-relevant frames (objects, via-points, attractors, etc.) and ii) Gaussian *joints* representing invariant geometric features between neighboring Gaussians. Given a new task instance, the Elastic-GMM is transformed using Laplacian Editing, generating novel task parameters used to re-estimate the LPV-DS policy. Elastic-DS is compositional in nature and can be used to construct flexible multi-step tasks. We showcase its strength on a myriad of simulated and real-robot experiments while preserving desirable control-theoretic guarantees.

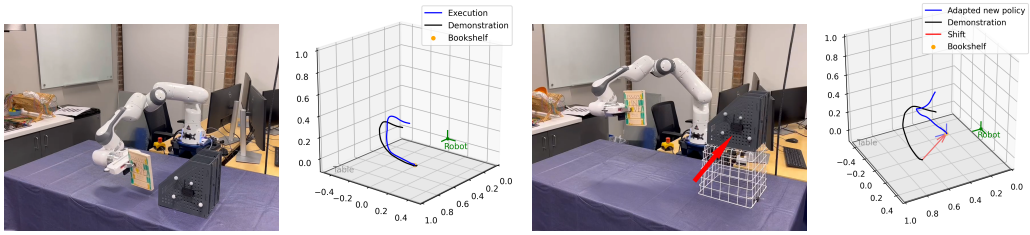
Keywords: Stable Dynamical Systems, Reactive Motion Policies, Learning from Demonstrations, Transfer Skill Learning, Task Transfer, Task Generalization

1 Introduction

With advanced development in robotics and autonomous systems in the past decades, the opportunities and demands for more complex physical human-robot interaction (pHRI) in our everyday unconstrained environments are rising; thus, it is critical for robots to be adaptive, compliant, reactive, safe and easy to program [1, 2, 3]. In many cases, robots will need to acquire new skills to satisfy task requirements in an ever-changing environment. It is usually difficult for non-experts to program robots for complex motion tasks and even tedious for experts to reprogram them when task requirements change. A more straightforward medium is needed for humans to convey intent to robots. An intuitive approach for robots to develop new skills is through Learning from Demonstration (LfD) [4, 5, 6, 7, 8], also known as imitation learning (IL) or programming by demonstration. This paradigm allows robots to acquire skills, typically encoded or defined in literature as action policies, motion policies, or imitation policies, directly from motion examples provided by humans or even other robots, mirroring a teacher-student relationship. As opposed to Reinforcement Learning (RL) [9], where the robot learns from multiple trial-and-errors and requires heavy reward design; LfD narrows down the policy search space, preventing learning from scratch or manually designing reward functions. It reduces the load and effort of humans, making it more suitable for pHRI.

In recent years, significant progress has been made in using LfD to learn complex and diverse motion tasks ranging from controlling Atari game agents to humanoid robots. However, many of them focused on learning from single tasks and executing the same tasks in the same scenario/environment. In reality, humans tend to borrow knowledge from prior experience of skill acquisition and reuse

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.



(a) Executing the learned DS (b) Trajectory for the original learned policy (c) Generalize the learned policy to new configuration (d) Trajectory for the new policy

Figure 1: Task-level transfer for bookshelf stacking with Elastic-DS. (a-b) Given a single Demonstration, the Elastic-DS can efficiently learn and reproduce the task, (c-d) and efficiently transfers to shifted and rotated task parameters. Our Elastic-DS approach transfers seamlessly and efficiently to new task parameter configurations while retaining stability and convergence guarantees.

them for acquiring and executing new tasks more efficiently. Instead of memorizing human demonstrations for one scenario, the robot should intelligently understand the objective, which means it should still achieve satisfactory performance in novel scenarios if the task objective is the same. From the machine learning perspective, this is the covariate shift issue that exists in many supervised learning tasks, including LfD methods, especially in the behavior cloning (BC) approach [7]. By providing a fixed training dataset beforehand, the LfD algorithm will learn a policy that performs well for the training dataset but could fail to generalize to unseen input during deployment. The learned policy will become invalid due to the change of distribution.

The Trilemma - ***Generalization*** or adaptation/transfer abilities are particularly important to enable robots to perform effectively in dynamic environments. There are many attempts at generalization with methods like BC [10], Inverse RL (IRL) [11, 12, 13], Meta-Learning [14], Multi-Task Learning [15], Transfer Learning [16], Multi-Task Reinforcement Learning [17], Lifelong Learning [18, 19, 20], and continual learning [21]. However, the aforementioned approaches have no emphasis on providing *control-theoretical guarantees* on the learned policies, such as stability, boundedness and convergence, all if which are critical for safe pHRI. On the other hand, the Dynamical System-based (DS) motion policy approach [3] offers many advantages such as reactivity, motion-level adaptation, and, most importantly, ***stability guarantees***; and can be learned from only handful of demonstrations ensuring ***minimal human effort*** [22, 23]. However, due to the closed-form and offline learning nature of DS-based motion policies, they have no flexibility for transfer to novel environments as they ignore explicit task parameters that inherently change the underlying trajectories that shape the DS vector fields. This limits their generalization capability and adoption as low-level policies for end-to-end, multi-task transfer. Hence, invoking the no free lunch theorem, we posit that the state-of-the-art currently suffers from the ***generalization vs. stability vs. effort trilemma***.

Goal In this work, we seek to alleviate this trilemma, by proposing an LfD approach that benefits from i) stability guarantees, ii) the flexibility to transfer skills across novel scenarios, while ii) requiring minimal human effort during learning and transfer, as depicted in Figure 1.

Related Work The number of techniques that exist for LfD/IL is vast [6, 4, 7, 8]. This work follows the BC approach, which learns a policy that maps states (state-action pairs, trajectories, and other contexts are also used) to control inputs [24]. DAgger [25] addressed distribution shift issues with online interactions/corrections. Whereas the generative adversarial learning framework [26] randomly explores for corrections that bring the policy close to the demonstrated distribution. These approaches offer generalization in terms of distribution shift, but require either constant human effort or lots of data and computation, and hold no control-theoretic guarantees on the learned policy. The recently introduced TaSIL (Taylor Series IL) framework introduces a simple augmentation to the BC loss such that the trained policy is robust to distribution deviations by ensuring incremental input-to-state stability, also benefiting from reduced sample-complexity [27]. Nevertheless, it cannot transfer to novel task instances/environments not seen during training. A Probably Approximately Correct (PAC)-Bayes IL framework was introduced in [10] that computes upper bounds on the expected cost of policies in novel environments. Prior works also focus on explicit skill transfer for novel

environments or task instances, such as multitasking learning [28, 29, 30, 31], meta-learning [14, 32]. While capable of generalizing learned tasks to different environments, all these works require a considerable amount of offline and online training and require excessively large DNNs which cannot be used in a reactive manner nor offer any form of control-theoretic guarantees.

A significant body of work tackles the generalization problem from a motion policy parameter perspective; such as transferring explicit start and goal positions in movement primitives [33], conditioning on probability distributions like the Probabilistic Movement Primitives (ProMPs) for different via-points [34] and geometric descriptor [35]. Other works emphasize task parametrization (TP) as relevant task frames in $SE(3)$ assigned to relevant objects in demonstrations, like TP-GMM [36, 37], TP-DMP [38], task invariants [39, 40] and environmental constraints [41]. While such works have demonstrated the ability for task generalization, few of them provide real-time reactive motion and stability guarantees, and most of them rely on the availability of demonstrations in different contexts or environments in order to extract the relevant task parameters used for transfer.

Approach We propose a zero-shot approach for transferring motion policies to novel scenarios, while guaranteeing control-theoretic properties for safe deployment in pHRI. To achieve *stability*, we adopt the DS-based LfD paradigm [3] that learns motion policies as time-invariant nonlinear DS with Lyapunov stability guarantees. To achieve *transfer*, we follow the task-parametrization perspective [36, 37, 38] and propose to embed relevant task frames directly into the DS policy. While several neural network (NN) based formulations for stable DS motion policies exist, such as neurally imprinted vector fields [42], DNN via contrastive learning [43], diffusion models [44], and euclideanizing flows [45]; given their black-box nature it is not straightforward to embed such task parameters into these formulations. Further, NN approaches need multiple demonstrations to properly encode the stable DS. To achieve *minimal data, compute and human effort* during learning, we adopt the Gaussian Mixture Model (GMM) based Linear Parameter Varying (LPV-DS) formulation [22, 3] which has shown to be computationally efficient and capable of learning stable vector fields of complex motions from single demonstration [23]. Finally, for *efficient transfer*, we take inspiration from elastic bands [46] and trajectory editing [47] and propose a novel approach to transfer the GMM-based LPV-DS to novel scenarios without new demonstrations, referred to as Elastic-DS.

Contributions We introduce the Elastic-DS formulation as a solution to the LfD trilemma (See Figure 2). The Elastic-DS is constrained to a set of task parameters described as geometric descriptors representing the invariant features of a task (e.g., object, via-point or target configurations). It is capable of efficiently generating novel DS policies upon task parameter changes without requiring new data or human input for single, multi-step tasks and composition of new tasks via DS stitching.

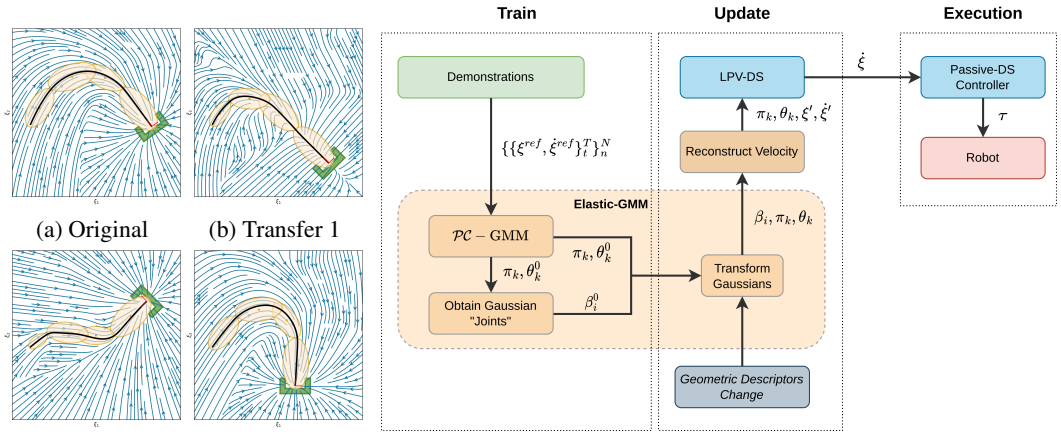
2 Problem Statement

Let $\mathcal{D} := \{\{\xi_{t,n}, \dot{\xi}_{t,n}\}_{t=1}^{T_n}\}_{n=1}^{N}$ be a set of N demonstration trajectories collected from kinesthetic teaching for a task, where $\xi_t \in \mathbb{R}^d$, $\dot{\xi}_{t,n} \in \mathbb{R}^d$ represent the kinematic robot state and velocity vectors at time t, respectively, for the n-th trajectory with length T_n. In this work, we consider $\xi_t \in \mathbb{R}^d$ to be the end-effector Cartesian position. Let $\dot{\xi} = f(\xi)$ be a first-order DS that describes a motion policy in $\mathbb{R}^n$ state space. Given $\mathcal{D}$, the goal is to infer $f(\xi) : \mathbb{R}^n \to \mathbb{R}^n$ such that any point ξ in the state space leads to a stable attractor ξ^*, with $f(\xi)$ described by a set of parameters Θ and ξ^*

$$\dot{\xi} = f(\xi; \Theta, \xi^*) \Rightarrow \lim_{t\to\infty} \|\xi - \xi^*\| = 0. \tag{1}$$

Usually, such DS motion policies are learned in an offline manner and fixed for the execution phase [23]. The main contribution of this work is to introduce an update stage that parametrizes the system further, allowing the motion policies to adapt and generalize to new tasks as shown in Figure 2.

Task Parameters A task is defined as a combination of multiple trajectories where each of them is grounded on a geometric constraint descriptor $O_i = \{o_{enter}, o_{exit}\} \in SE(3)^2$, describing the two endpoint poses. Let $\beta_i \in \mathbb{R}^{d\times(k+1)}$ be key way-points in the state space conditioned on the geometric descriptors O_i. As O_i changes, the task parameters β_i will change accordingly to generate a new set of DS to reach O_i with the correct poses. In this work, the geometric descriptor O_i is assumed to be known or given during demonstrations. However, it could come from various upstream sources such as human specifications [23] or generative segmentation algorithms [48].



(a) Original (b) Transfer 1 (c) Transfer 2 (d) Transfer 3 (e) Elastic-DS Learning and Control Flowchart

Figure 2: (a) Elastic-DS reproducing a stable LPV-DS vector field on the originally demonstrated data. (b-d) Elastic-DS generating stable LPV-DS motion policies from peg attractor changes (translation and rotation) without new demonstration. (e) Flowchart of Elastic-DS approach.

Motion Policy We propose the following motion policy formulation for task parameter transfer:

$$\dot{\xi} = \sum_{i=1}^{M} \delta(\xi, O_i) f_i(\xi; \Theta_i, \beta_i, \xi_i^*) \tag{2}$$

where $\delta(\xi, O_i)$ is an activation function determining the sequence of execution for M DSs describing a multi-step sequential tasks. Hence, given $\mathcal{D}$ with the same behavior and M geometric descriptor O_i configurations, our approach finds the task parameters β_i that will generate new DS motion policies with i) stability guarantees with respect to their corresponding attractors ξ_i^*, and ii) the flexibility to achieve the same task with new geometric constraint descriptor configurations.

3 Preliminaries: $\mathcal{PC}$-GMM and LPV-DS Motion Policy [22]

The GMM-based LPV-DS [22] motion policy has the following formulation,

$$\dot{\xi} = f(\xi) = \textstyle\sum_{k=1}^{K} \gamma_k(\xi) \left(A^k \xi + b^k\right) \quad \text{s.t.} \quad \begin{cases} \left(A^k\right)^T P + P A^k = Q^k, Q^k = \left(Q^k\right)^T \prec 0 \\ b^k = -A^k \xi^* \end{cases} \tag{3}$$

where $\gamma_k(x)$ is the state-dependent mixing function that quantifies the weight of each linear time-invariant (LTI) system $(A^k \xi + b^k)$. $\mathcal{N}(\xi|\theta_k)$ denotes the probability of observation ξ from the k-th Gaussian component parametrized by $\theta_k = \{\mu_k, \Sigma_k\}$, and π_k represents the prior probability of an observation from this particular component and the a posteriori probability is

$$\gamma_k(\xi) = \frac{\pi_k \mathcal{N}(\xi|\theta_k)}{\sum_{j=1} \pi_j \mathcal{N}(\xi|\theta_j)} \quad \text{from} \quad p\left(x|\{\pi_k, \theta_k\}\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\xi|\mu_k, \Sigma_k). \tag{4}$$

Intuitively, Eq. 3 fits a mixture of linear DS to a complex non-linear trajectory, with $\gamma_k(x)$ ensuring the smoothness of the reproduced trajectories. Hence, it requires for each Gaussian component to be placed on quasi-linear segments of $\mathcal{D}$.

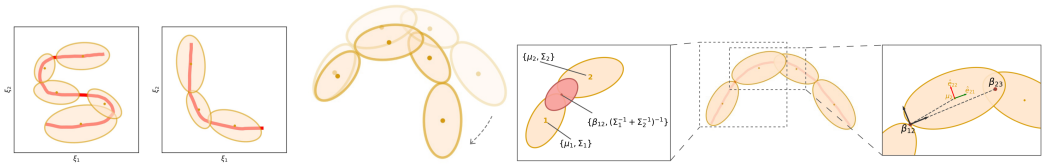
$\mathcal{PC}$-GMM To achieve such locally linear partitioning of $\mathcal{D}$ the Bayesian non-parametric physically consistent $\mathcal{PC}$-GMM was introduced in [22]. By imposing a prior on the Gaussian component assignments, biased by a similarity metric that leverages locality and directionality, the $\mathcal{PC}$-GMM ensures that the datapoints assigned to each k-th component correspond to a linear DS. Further, the number of Gaussians K is automatically estimated through a collapsed Gibbs sampling approach. Hence, the optimal number of linear DS A^k for a given $\mathcal{D}$ can be automatically inferred.

Stability To guarantee global asymptotic stability of Eq. 3, a Lyapunov function $V(x) = (x - x^*)^T P (x - x^*)$ with $P = P^T \succ 0$, is used to derive the stability constraints in Eq. 3. Minimizing the fitting error of Eq. 3 with respect to demonstrations subject to constraints in Eq. 3 yields a non-linear DS with a stability guarantee via a Semi-Definite Program (SDP) [22]. Implementation details are provided in Appendix A.

4 Elastic-DS

4.1 Elastic-GMM

To generate the task parameters β_i for the corresponding spatial change in the geometric descriptors $\mathcal{O}_i$, we introduce Elastic-GMM as the core component of augmenting LPV-DS (Eq. 3) into Elastic-DS (Eq. 2). Figure 2e shows the pipeline of Elastic-DS. During the training stage, we use $\mathcal{PC}$-GMM [22] to obtain a set of initial Gaussian parameters $\{\pi_k, \theta_k^0\}$ as well as the initial task parameter β_i^0. The update stage will produce the updated Gaussian parameters $\{\pi_k, \theta_k\}$ as well as the updated task parameter β_i, which are key way-points in the state space. A trajectory will be generated based on the key points to specify the velocity for the DS after the update. With the new Gaussian parameters and the new velocity information, an updated LPV-DS can be learned. If further updates happen in the environment, we can directly update LPV-DS using the transform without re-estimating the GMM, which avoids a time-consuming stage. During the execution, a passive-DS controller [3] will take the newest LPV-DS output velocity $\dot{\xi}$ to generate the corresponding joint torque τ for the robots. The upcoming section will discuss each key component in detail.



(a) GMM from trajectories (b) GMM Chain (c) Gaussian *joints* illustration in 2D

Figure 3: Illustrations of $\mathcal{PC}$-GMM fitted on trajectory data (a) and the Gaussian chain transform from Section 4.1.1, (c) a closer look at the Gaussian *joints* introduced in Section 4.1.3.

4.1.1 GMM Chain

Following the LPV-DS pipeline, we first turn the demonstration trajectory into clusters described by a GMM $\{\pi_k, \theta_k^0\}$ using $\mathcal{PC}$-GMM [22]. As shown in Figure 3a, a trajectory is extracted and simplified into a *chain of Gaussians links*. In the update stage (Figure 2e), we utilize the transformation of the spatial relationships among the Gaussians to achieve task adaptation. Imagine an analogy of the Gaussian chain being a robot arm, which could be rotated around each joint to achieve a specific geometric configuration as shown in Figure 3b. Note the robot arm analogy is on the end-effector trajectory instead of the actual robot arm. The task parameters β_i are the *joints* between each pair of the neighboring Gaussians, which describes the spatial relationship between the neighbors. After the $\mathcal{PC}$-GMM step, we can obtain the initial β_i^0 and later update the β_i^0 to β_i to achieve transform. The *joint* of two Gaussian, which is also the task parameter β_i, is the mean of the product between them as described by the picture on the left in Figure 3c, which could be obtained by,

$$\begin{aligned} \Sigma_t &= (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1} \\ \beta_{i,12} &= \Sigma_t(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2) \end{aligned} \tag{5}$$

where μ and Σ are the mean and covariance of the Gaussians $\{\pi_1, \theta_1^0\}$ and $\{\pi_2, \theta_2^0\}$. To complete the robot arm analogy, we also need to determine the *links* position and orientation $T_{GMM} \in SE(3)^K$ with respect to the *joints*. Figure 3c depicts the Gaussian mean position μ_j and orientation (described by the eigenvectors $\hat{e}_j$ of the covariance matrix Σ_j) with respect to the frame at the last joint with the x-axis pointing towards the next joint (in the direction of the demonstration). All of the above are constructed as the initial condition, in which no update is involved yet. Later when the states of the task parameter β_i change, we will recover the same transformation of the mean and covariance with respect to the corresponding β_i. Before introducing the approach to obtain the new *joint* positions β_i we provide a brief summary of the Laplacian editing approach [49, 47].

4.1.2 Laplacian Editing Primer

Laplacian Editing allows directly modifying an existing trajectory defined by m waypoints $\boldsymbol{r} \in \mathbb{R}^{d\times m}$ while capturing local properties. First, we convert the waypoints $\boldsymbol{r}$ in cartesian space into

Laplacian coordinates δ with the discrete Laplace-Beltrami operator for each waypoint r_i [47],

$$\delta_i = \sum_{j \in \mathbf{N}_i} \frac{w_{ij}}{\sum_{j \in \mathbf{N}_i} w_{ij}} (r_i - r_j) \quad \text{and} \quad M_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\dfrac{w_{ij}}{\sum_{j \in \mathbf{N}_i} w_{ij}} & \text{if } j \in \mathbf{N}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

where $\mathbf{N}_i$ are a set of neighbor points r_j for waypoint r_i, and w_{ij} is a weight set to 1 for this work. By concatenating all the δ_i as $\boldsymbol{\delta}$, it can be decomposed into $M\boldsymbol{r} = \boldsymbol{\delta}$ where $M \in \mathbb{R}^{m \times m}$. The matrix M can be singular, so one can impose constraints on the system $M\boldsymbol{r} = \boldsymbol{\delta}$ when solving for new waypoints $\boldsymbol{r}$ to achieve editing [47].

4.1.3 Transform Gaussians with Constraints

We first connect all the joints β_i together with the demonstration start and end point to form a piecewise linear trajectory link. Then, we align the first and the last link with the geometric descriptor $\mathcal{O}_i$ as the constraints. For the objective, the initial β_i^0 is converted into the Laplacian coordinate and we construct a least-square formulation as in Section 4.1.2. When solving for this optimization, the other joint positions will adjust based on the Laplacian objective, softly preserving local properties,

$$\begin{aligned} &\min_{\beta_i} J(\beta_i) = \|L\beta_i - \Delta\|_2^2 \\ &\text{subject to} \begin{cases} T_{0,1}(\beta_{i,0}, \beta_{i,1}) = O_{start} \\ T_{n-1,n}(\beta_{i,n-1}, \beta_{i,n}) = O_{end} \end{cases} \end{aligned} \tag{7}$$

where $L \in \mathbb{R}^{n \times n}$ is the graph Laplacian matrix, Δ is the Laplacian coordinates matrix transformed from β in the Cartesian coordinates. $T_{0,1}(\beta_0, \beta_1)$ represents the transformation from $\beta_{i,0}$ to $\beta_{i,1}$. The solution of this optimization will produce new *joints* positions β_i. After that, the *link* position and orientation (which are the Gaussians' means and covariances), as well as the scale, are recovered using the T_{GMM} recorded from the previous section. The scale is determined by the change from the original distance between each pair of neighboring *joints*. With the new distances between the *joints*, we then scale the Gaussians accordingly by scaling the eigenvalues of the Gaussians' covariances. Referring back to the flowchart in Figure 2e, this section outputs the updated *joint* positions β_i and updated GMM parameters θ_k (π_k stay unchanged).

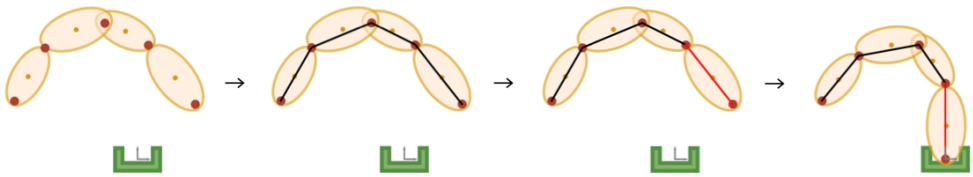
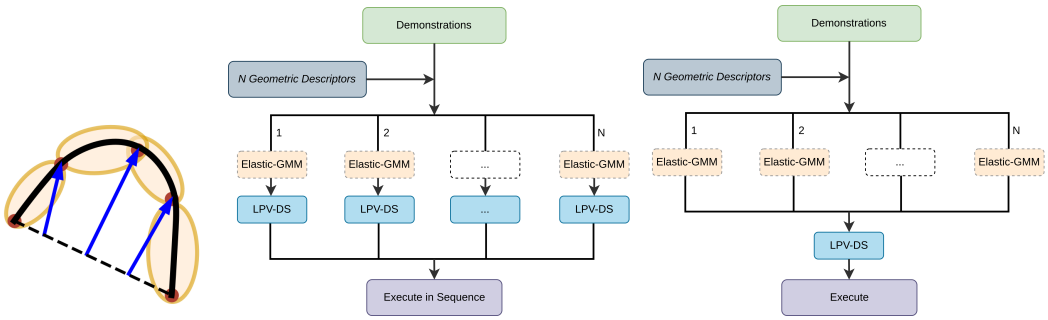


Figure 4: After obtaining the joints between neighboring Gaussians, a piecewise linear trajectory is formed. The green polygon is the geometric descriptor at the exit of the trajectory. We set a constraint on the last segment to align with the pose of the geometric descriptor. After solving (7), we achieve the final transformation (depicted in the rightmost image). Note: this illustration only shows the constraint at the exit, while in general, the constraints could be at the entry and the exit.

4.2 Create Velocity Profile

Depending on the new task constraints, the velocity requirement could be different from the original demonstration. Therefore, this approach offers the opportunity to modify the velocity by regenerating a trajectory along the Gaussians *joints* as the waypoints. There are many ways to achieve this with known waypoints, such as splines or minimum jerk trajectory. We provide a simple example of using Laplacian Editing to generate a new trajectory. First, we connect $\beta_{i,0}$ and $\beta_{i,n}$ to form a linear trajectory $\zeta \in \mathbb{R}^{d \times p}$. p is the number of points on this trajectory. We then force this trajectory to pass through β, the Gaussian joints, with Laplacian editing,

$$\begin{aligned} &\min_{\zeta} J(\zeta) = \|L_\zeta \beta - \Delta_\zeta\|_2^2 \\ &\text{subject to } \zeta_j = \beta_{i,q} \end{aligned} \tag{8}$$



(a) Connect endpoints and set constraints

(b) This flow ensures that the position geometric constraints will be met for the task

(c) This flow produces smooth motion but could have less task-satisfaction

Figure 5: (a) An illustration of generating trajectory by modifying from a linear trajectory. This example includes added interpolation points at the first and the last segments to help align with the endpoint constraints. (b-c) Different task split flows for design trade-off.

where $j \in \{0, ..., p-1\}$ is the index in ζ for matching the corresponding β_i. We first calculate the total Euclidean distance along the piecewise connected line between neighboring *joints* with total distance $D = \sum_{i=0}^{n-1} ||\beta_{i+1} - \beta_i||$. Then we have the percentage λ_q of the progress that the *joints* positions in β_i make along the total distance D. By using λ_q, the corresponding β_i is mapped to the index $j = \lfloor \lambda_q(p-1) \rfloor$. In practice, by also enforcing constraints between the *joints* with linearly interpolation will help the trajectory become more aligned with the GMM link and the geometric descriptors. The velocity will be determined by the finite difference between the edited trajectory neighboring data points divided by the dt collected from the demonstrations. By controlling the spacing between the edited trajectory neighboring data points and the number of points p, one can specify the velocity. After this section, we have the velocity information as well as the updated GMM from the previous section. They will become the input for learning a new DS motion policy following Section 3, which by nature preserves the stability guarantees.

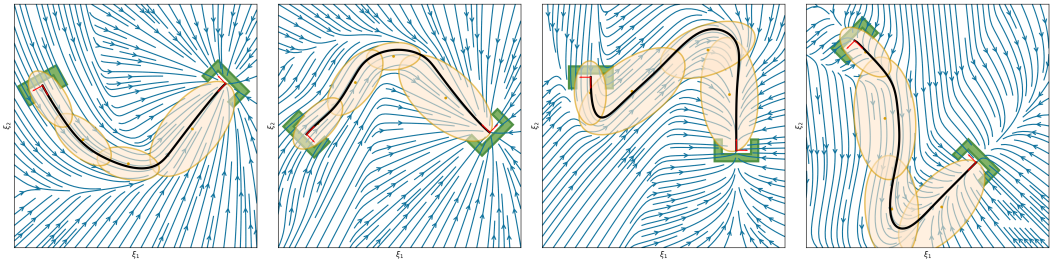
4.3 Task Split

The previous sections only introduce the scenario in which a task is described by a single trajectory with spatial constraints on both ends. The index i in the task parameters β_i indicates that there could be multiple segments of such transforms, consequently multiple DSs in (2) when $M > 1$. To allow adding spatial constraints in the middle, one can split the task into multiple segments and process them in a divide-and-conquer manner. The activation function $\delta(\xi, O_i)$ is in charge of the switch among the multiple learned DS cases. The switch is simply when one DS reaches its own attractor, δ will start to activate the next DS following the demonstration order. Such pipeline is shown in Figure 5b. Figure 5c shows the other case, which is to connect all the GMMs from different segments and learn a single DS. The δ function is not in use for this case. As mentioned in Section 2, the interesting separation points described by the geometric descriptors are specified by some upstream sources. For more information about the split and stitching process, please refer to Appendix C. By breaking up trajectories, they could be regrouped and reused for new tasks. The flexibility of composing different transformed DS with the new constraint poses allows the possibility for multi-task scenarios.

5 Experimental Results

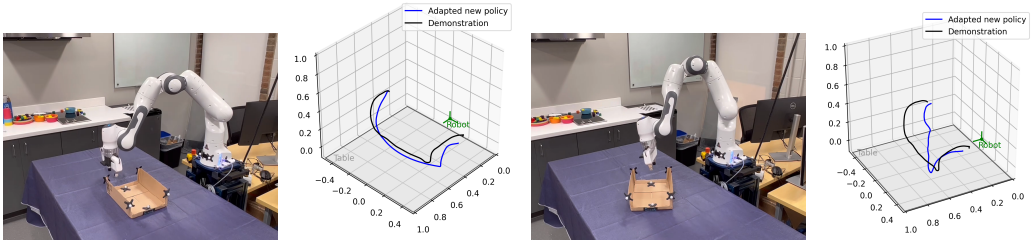
5.1 2D Experiments

This section shows 2D examples of using Elastic-DS. The learned DS is plotted as steamplots describing a velocity vector field (in blue) with GMM (in orange) geometric descriptors (in green) and rollout trajectory (in black) overlaying on top. The 2D simulation in Figure 6 shows the atomic case of one segment trajectory conditioned on a geometric descriptor with constraints at the endpoints. The two ends of the geometric descriptor (green polygons) are shifted and rotated to show different configurations and the changes in the DS vector field. The DS motion policy is able to adapt to the changes. For more 2D experiments with task split, please refer to the Appendix.



(a) The original DS (b) Flip (c) Change to an S shape (d) Reverse checkmark

Figure 6: Elastic-DS: DS motion policy transfer based on changes of start and goal poses



(a) Executing the learned DS (b) Trajectory for the original learned policy (c) Generalize the learned policy to a rotated tunnel (d) Trajectory for the new policy

Figure 7: Example of Elastic-DS adaptation in a tunnel passing task

5.2 Robot Experiments

We demonstrate the validation result through four different real robot experiments on the Franka Emika Panda robot: Bookshelf, Pick and Place, Tunnel, and Combination, see Figure 2, 7 and Appendix D. In these experiments, the geometric descriptors are detected from a motion capture system. By attaching motion capture markers on objects, we specify geometric descriptors anchored on the objects of interest. Three experiments will start with a human performing kinesthetic teaching by moving the end-effector. Then, the execution of the original learned DS from the demonstration will be shown. The objects of interest will then be shifted and rotated with different configurations. The last experiment shows the ability to compose new tasks. Without any new demonstration, the robot can still achieve the required tasks. Please refer to the video and Appendix D.

6 Conclusions and Limitations

We propose Elastic-DS in this work, which allows modifying DS with task parameters conditioned on geometric features to achieve task generalization. As the core component, we introduced Elastic-GMM to augment the original LPV-DS to create more flexible task-specific motions with as low as one demonstration. By showing both 2D simulation and different 3D robot experiments, we validate the ability of Elastic-DS to perform task generalization as well as the potential for multi-task and long-horizon motion policies. Following we discussed the limitations of our approach.

Limitations First, this work only considers end-effector motions in the Cartesian position space. The task constraints are only for translational motion as well. To achieve more variety of tasks and extend to more possible poses, the orientation space has to be considered. Further directions could adopt works like [50] and [51] to produce DS motion policies and meet task constraints in the full pose space. To go even further, the full pose could also contain the gripper state, which could be achieved with a coupled DS approach such as [52] and [3]. Motion policy in the joint space should also be considered [53]. Second, the geometric descriptors are assumed to be given by human specification in this work. To address this limitation, we could utilize methods like Neural Radiance Fields [54] to identify the object of interest which the demonstrated trajectories pass through. Finally, the task adaptation in this work is not real-time (1 to 5 seconds on a typical laptop, depending on the complexity of the task). With the dynamic nature of physical human-robot interaction, it is important to provide continuous adaptation on the fly to create a seamless experience.

References

[1] P. A. Lasota, T. Fong, and J. A. Shah. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5(4):261–349, 2017. ISSN 1935-8253.

[2] L. Sanneman, C. Fourie, and J. A. Shah. The state of industrial robotics: Emerging technologies, challenges, and key research directions. *Foundations and Trends® in Robotics*, 8(3): 225–306, 2021. ISSN 1935-8253.

[3] A. Billard, S. Mirrazavi, and N. Figueroa. *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. MIT Press, 2022.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[5] S. Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[6] A. G. Billard, S. Calinon, and R. Dillmann. Learning from humans. *Springer handbook of robotics*, pages 1995–2014, 2016.

[7] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[8] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3: 297–330, 2020.

[9] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[10] A. Ren, S. Veer, and A. Majumdar. Generalization guarantees for imitation learning. In *Conference on Robot Learning*, pages 1426–1442. PMLR, 2021.

[11] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[12] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. *Active preference-based learning of reward functions*. 2017.

[13] Z. Zhao, Z. Wang, K. Han, R. Gupta, P. Tiwari, G. Wu, and M. J. Barth. Personalized car following for autonomous driving with inverse reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2891–2897, 2022. doi: 10.1109/ICRA46639.2022.9812446.

[14] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[15] H. Wang, H. Zhao, and B. Li. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In *International Conference on Machine Learning*, pages 10991–11002. PMLR, 2021.

[16] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[17] L. Sun, H. Zhang, W. Xu, and M. Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=LYXTPNWJLr.

[18] S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15 (1-2):25–46, 1995.

[19] P. Ruvolo and E. Eaton. Ella: An efficient lifelong learning algorithm. In *International conference on machine learning*, pages 507–515. PMLR, 2013.

[20] J. A. Mendez, H. van Seijen, and E. Eaton. Modular lifelong reinforcement learning via neural composition. *arXiv preprint arXiv:2207.00429*, 2022.

[21] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019.

[22] N. Figueroa and A. Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 927–946. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/figueroa18a.html.

[23] Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=ndYsaoyzCWv.

[24] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[25] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[26] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[27] D. Pfrommer, T. T. Zhang, S. Tu, and N. Matni. TaSIL: Taylor series imitation learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=jqzoJw7xamd.

[28] K. Zentner, U. Puri, Y. Zhang, R. Julian, and G. S. Sukhatme. Efficient multi-task learning via iterated single-task transfer. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10141–10146. IEEE, 2022.

[29] P. K. Mudrakarta, M. Sandler, A. Zhmoginov, and A. Howard. K for the price of 1: Parameter-efficient multi-task and transfer learning, 2019.

[30] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[31] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022.

[32] H. Fu, S. Yu, S. Tiwari, G. Konidaris, and M. Littman. Meta-learning transferable parameterized skills. *arXiv preprint arXiv:2206.03597*, 2022.

[33] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[34] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

[35] N. Freymuth, N. Schreiber, P. Becker, A. Taranovic, and G. Neumann. Inferring versatile behavior from demonstrations by matching geometric descriptors. *arXiv preprint arXiv:2210.08121*, 2022.

[36] S. Calinon. Robot learning with task-parameterized generative models. In *Robotics Research: Volume 2*, pages 111–126. Springer, 2017.

[37] J. Zhu, M. Gienger, and J. Kober. Learning task-parameterized skills from few demonstrations. *IEEE Robotics and Automation Letters*, 7(2):4063–4070, 2022.

[38] A. Pervez and D. Lee. Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11(1):61–78, 2018.

[39] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard. Task parameterization using continuous constraints extracted from human demonstrations. *IEEE Transactions on Robotics*, 31(6):1458–1471, 2015. doi:10.1109/TRO.2015.2495003.

[40] N. Figueroa, A. L. P. Ureche, and A. Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 611–612, 2016. doi:10.1109/HRI.2016.7451881.

[41] X. Li and O. Brock. Learning from demonstration based on environmental constraints. *IEEE Robotics and Automation Letters*, 7(4):10938–10945, 2022.

[42] K. Neumann, A. Lemme, and J. J. Steil. Neural learning of stable dynamical systems based on data-driven lyapunov candidates. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1216–1222. IEEE, 2013.

[43] R. Pérez-Dattari and J. Kober. Stable motion primitives via imitation and contrastive learning. *arXiv preprint arXiv:2302.10017*, 2023.

[44] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[45] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, and N. Ratliff. Learning reactive motion policies in multiple task spaces from human demonstrations. In *Conference on Robot Learning*, pages 1457–1468. PMLR, 2020.

[46] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2, 1993. doi:10.1109/ROBOT.1993.291936.

[47] T. Nierhoff, S. Hirche, and Y. Nakamura. Spatial adaption of robot trajectories based on laplacian trajectory editing. *Autonomous Robots*, 40:159–173, 2016.

[48] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[49] Y. Lipman, O. Sorkine, M. Alexa, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Laplacian framework for interactive mesh editing. *International Journal of Shape Modeling*, 11(01):43–61, 2005.

[50] N. Figueroa, S. Faraji, M. Koptev, and A. Billard. A dynamical system approach for adaptive grasping, navigation and co-manipulation with humanoid robots. In *2020 IEEE International conference on robotics and automation (ICRA)*, pages 7676–7682. IEEE, 2020.

[51] J. Urain, D. Tateo, and J. Peters. Learning stable vector fields on lie groups. *IEEE Robotics and Automation Letters*, 7(4):12569–12576, 2022.

[52] A. Shukla and A. Billard. Coupled dynamical system based arm–hand grasping model for learning fast adaptation strategies. *Robotics and Autonomous Systems*, 60(3):424–440, 2012.

[53] S. M. Khansari-Zadeh and O. Khatib. Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots*, 41:45–69, 2017.

[54] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.