# Zero-Shot Generalization from Motion Demonstrations to New Tasks

Kilian Freitag[1*], Alvin Combrink[1*], and Nadia Figueroa[2]

***Abstract*— Learning motion policies from expert demonstrations is an essential paradigm in modern robotics. While end-to-end models aim for broad generalization, they require large datasets and computationally heavy inference. Conversely, learning dynamical systems (DS) provides fast, reactive, and provably stable control from very few demonstrations. However, existing DS learning methods typically model isolated tasks and struggle to reuse demonstrations for novel behaviors. In this work, we formalize the problem of combining isolated demonstrations within a shared workspace to enable generalization to unseen tasks. The *Gaussian Graph* is introduced, which reinterprets spatial components of learned motion primitives as discrete vertices with connections to one another. This formulation allows us to bridge continuous control with discrete graph search. We propose two frameworks leveraging this graph: *Stitching*, for constructing time-invariant DSs, and *Chaining*, giving a sequence-based DS for complex motions while retaining convergence guarantees. Simulations and real-robot experiments show that these methods successfully generalize to new tasks where baseline methods fail.**

## I. Introduction

Learning from demonstration is a widely used paradigm in robotics for acquiring control policies from expert trajectories. Recent work increasingly targets end-to-end foundation models that aim to generalize across tasks and embodiments [1], [2]. However, such models typically require large datasets and significant compute, which can make fast, reactive control challenging [3].

An alternative is to learn a dynamical system (DS) from a small number of demonstrations and use it directly as a motion policy [4], [5]. In *Linear Parameter Varying DS* (LPV-DS) [4], a Gaussian Mixture Model (GMM) partitions the demonstrations into regions and smoothly blends locally linear time-invariant (LTI) DSs into a single nonlinear policy, enabling stable and convergent behavior toward a target state. Despite their data efficiency, LPV-DS-like methods are typically trained per task. Consequently, they do not generalize to new initial-target pairs that were not demonstrated.

The ability to combine demonstrations of distinct tasks to solve novel, unseen tasks has high practical relevance in modern robotics. In structured environments such as factories or laboratories, the number of potential tasks scales combinatorially with the number of start and goal configurations. Collecting expert demonstrations for every possible variation is prohibitively expensive, time-consuming, and fundamentally unscalable. However, demonstrations in a shared workspace effectively exhibit *regions of safe and kinematically viable movement* — knowledge that we aim to repurpose in this work to achieve zero-shot generalization to new tasks. This approach drastically improves the efficiency and scalability of imitation learning by circumventing the need for exhaustive data collection. Furthermore, it naturally supports incremental learning; introducing a new workstation to a factory floor only requires a brief demonstration linking it to the existing network of learned behaviors.

Prior work has explored the adjacent problem of adapting learned policies to new target configurations without requiring new data. For instance, *Elastic Motion Policy* [6] geometrically reshapes the underlying GMM structure to reach new targets, while *Task-Parameterized GMMs* [7] warp a single demonstrated skill to accommodate new or moving reference frames. Other approaches address multi-stage behaviors by segmenting demonstrations into sequential phases and recombining them temporally [8]. However, these methods primarily focus on adapting or segmenting an isolated task; they do not treat disjoint demonstrations as a shared, interconnected map of safe, kinematically viable regions from which entirely novel paths can be routed.

Building on LPV-DS, we reinterpret the Gaussians derived from the GMM as intermediate "stepping stones" that guide the system from its current state $\xi$ toward a target state $\xi^*$. By constructing a graph where these Gaussian centers serve as vertices, we can use established graph search algorithms to find paths that synthesize new motion policies from separate demonstrations, yielding efficient trajectories for unseen initial-target pairs in a zero-shot manner (see Fig. 1).

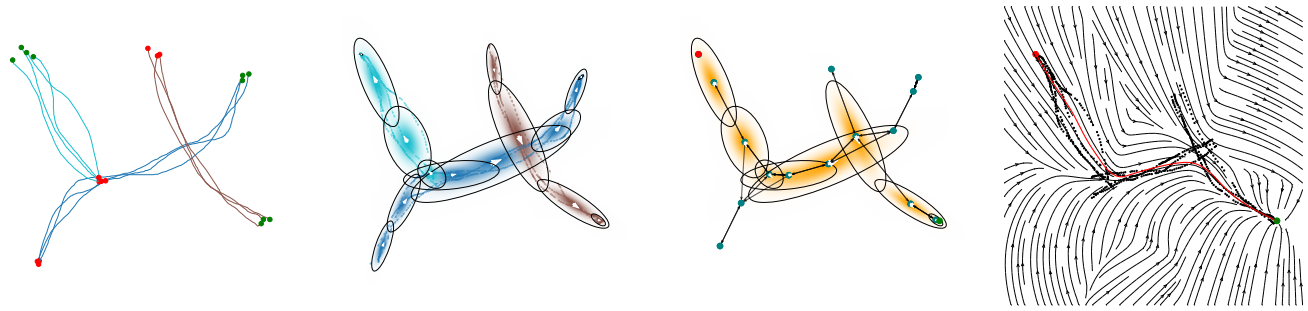
Our main contributions can be summarized as:

- The *Gaussian Graph* (GG), a novel representation combining multiple demonstrations and enabling the use of well-known graph algorithms for synthesizing new motion policies.
- *Demonstration Stitching*: GG-based construction of time-invariant DSs, enabling zero-shot initial-target generalization within data support and providing strict convergence guarantees.
- *Demonstration Chaining*: A sequential GG-based DS producing sophisticated motions while maintaining the same generalization and convergence properties.
- An empirical evaluation demonstrating consistent generalization to unseen tasks and real-robot experiments verifying the method's capabilities.

Our implementation builds on LPV-DS [4] and is provided along with datasets and additional details on our repository[1].

*Equal contribution. [1]Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden {combrink,kilian.freitag}@chalmers.se. [2]GRASP Lab, University of Pennsylvania, PA 19104, USA. This work is supported by the ITEA project ArtWork.

[1]https://github.com/KilianFt/DemoStitching.



(a) Set of Demonstrations. (b) Demo-wise LPV-DS. (c) Gaussian Graph and solution. (d) Stitched DS.

Fig. 1: The *Demonstration Stitching* (Section IV) pipeline using *Shortest Path* and *Reuse Gaussians*. On a set of separate demonstrations **(a)** (initial positions in red, target positions in green), LPV-DS is applied to each demonstration separately **(b)**, from which a Gaussian Graph is constructed and a shortest path from an initial position to a target position is found **(c)**. A DS is constructed to follow the shortest path **(d)**. Thus, a zero-shot generalization from separate motion demonstrations to a new, undemonstrated task is achieved.

## II. Problem Description and Preliminaries

We consider a setting where demonstrations for several different tasks are collected in the same workspace. Each demonstration $\mathcal{D} = \{\xi_i^{ref}, \dot{\xi}_i^{ref}\}_i$ — with $\xi_i^{ref}, \dot{\xi}_i^{ref} \in \mathbb{R}^d$ — aggregates position-velocity pairs from multiple reference trajectories. Intuitively, each demonstration provides position-velocity pairs that specify the desired robot motion within a region of the workspace (see Fig. 1a). Together, the different demonstrations form a demonstration set $\mathbb{D}$, which therefore contains a combined understanding of the workspace. The aim of this work is to effectively reuse information from these separate demonstrations for zero-shot generalization to unseen tasks by synthesizing new, safe movements from existing trajectories (see Fig. 1d) without collecting large amounts of data.

LPV-DS learns a time-invariant DS-based motion policy $\dot{\xi} = f_\Theta(\xi)$ from a single demonstration $\mathcal{D}$. The policy $f_\Theta$ is parameterized by $\Theta$ and constructed to ensure convergence to a stable equilibrium point $\xi^* \in \mathbb{R}^d$. That is, $\lim_{t\to\infty} \|\xi - \xi^*\| = 0$ when starting from any position in $\mathbb{R}^d$ and moving according to $\dot{\xi} = f_\Theta(\xi)$. We say that $f_\Theta$ is *globally asymptotically stable* (GAS) with respect to an *attractor* $\xi^*$. This is done by formulating $f_\Theta$ as a GMM of $K$ LTI dynamical systems:

$$f_\Theta(\xi) = \sum_{k=1}^{K} \gamma_k(\xi)\mathbf{A}_k(\xi - \xi^*), \tag{1}$$

$$\gamma_k(\xi) = \frac{\pi_k \mathcal{N}(\xi \mid \mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\xi \mid \mu_j, \mathbf{\Sigma}_j)}, \tag{2}$$

where $\mathbf{A}_k \in \mathbb{R}^{d\times d}$ is the state matrix describing the dynamics of the $k^{\text{th}}$ LTI DS component, $\pi_k$ is the prior probability of an observation belonging to Gaussian $k$, and $\mathcal{N}(\xi \mid \mu_k, \mathbf{\Sigma}_k)$ (with mean $\mu_k$ and covariance $\mathbf{\Sigma}_k$) is the conditional probability density of observing $\xi$ at Gaussian $k$. The posterior probability $\gamma_k(\xi)$ of $\xi$ belonging to system $k$ therefore weighs $\mathbf{A}_k$'s influence on $f_\Theta$. From this we see that the model parameters are $\Theta = \{\theta_k^{DS}, \theta_k^{GMM}\}_{k=1}^K$, with $\theta_k^{DS} = \mathbf{A}_k$ and $\theta_k^{GMM} = \{\pi_k, \mu_k, \mathbf{\Sigma}_k\}$. To find these parameters, first a GMM model is fit to $\mathcal{D}$, giving $\theta_k^{GMM}$ for $k = 1, \ldots, K$. Then, for the DS parameters $\theta_k^{DS}$, a parameterized Lyapunov function $V(\xi) = (\xi - \xi^*)^T \mathbf{P}(\xi - \xi^*)$ with $\mathbf{P} = \mathbf{P}^T \succ 0$ is formulated. By ensuring that $\mathbf{A}_k^T\mathbf{P} + \mathbf{P}\mathbf{A}_k = \mathbf{Q}_k$ and $\mathbf{Q}_k = \mathbf{Q}_k^T \prec 0$ for all $k$, $f_\Theta$ is GAS with respect to $\xi^*$. $\mathbf{P}$ is found simultaneously with $\theta_k^{DS}$.

By parameterizing $V(\xi)$ with $\mathbf{P}$, LPV-DS achieves greater flexibility in the LTI systems $\mathbf{A}_k$ [9] compared to methods like SEDS [8], which uses the unparameterized version. However, LPV-DS — just like SEDS but to a lesser degree — remains constrained by $V(\xi)$ to guarantee GAS, and therefore struggles as the non-linearity of the underlying motions increases. Consequently, naively applying LPV-DS to aggregated demonstrations from multiple tasks causes the motion policy to fit the demonstration data less accurately as task diversity grows. Section V proposes a method that circumvents this constraint while still guaranteeing GAS.

## III. The Gaussian Graph

To aggregate multiple demonstrations, we introduce the *Gaussian Graph* (GG), which enables the construction of new motion policies for unseen tasks (see Fig. 1c).

First, LPV-DS is applied to each demonstration $\mathcal{D} \in \mathbb{D}$ to obtain motion policies $\{f_\mathcal{D} \mid \mathcal{D} \in \mathbb{D}\}$ (dropping the parameter $\Theta$ subscript), where each $f_\mathcal{D}$ has parameters $\Theta_\mathcal{D}$, $\mathbf{P}_\mathcal{D}$, and $K_\mathcal{D}$. The GG is a directed, weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, W \rangle$ with vertices $\mathcal{V}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge-weights $W : \mathcal{E} \to \mathbb{R}^+$. Each vertex in $\mathcal{V}$ corresponds to a component in the motion policies. Specifically, for every component $k \in [1, .., K_\mathcal{D}]$ of motion policy $f_\mathcal{D}$ with parameters $\theta_{k,\mathcal{D}}^{GMM} = \{\pi_{k,\mathcal{D}}, \mu_{k,\mathcal{D}}, \mathbf{\Sigma}_{k,\mathcal{D}}\}$ and $\theta_k^{DS} = \mathbf{A}_{k,\mathcal{D}}$, there exists a vertex $v_{k,\mathcal{D}} \in \mathcal{V}$. The *direction* $\psi_{k,\mathcal{D}}$ is defined by the dynamics at the Gaussian center: $\psi_{k,\mathcal{D}} = \mathbf{A}_{k,\mathcal{D}}(\mu_{k,\mathcal{D}} - \xi_\mathcal{D}^*)$, where $\xi_\mathcal{D}^*$ is the attractor of $\mathcal{D}$.

Informally, we assume the full system dynamics $f_\mathcal{D}$ near component $k$ strongly aligns with $k$'s direction, $f_\mathcal{D}(\mu_{k,\mathcal{D}}) \approx \psi_{k,\mathcal{D}}$. Under this assumption, dynamics near $\mu_{k,\mathcal{D}}$ lead

toward components in the direction of $\psi_{k,\mathcal{D}}$, with closer components exerting greater influence until their own dynamics eventually dominate. Based on this principle, the connection from vertex $v_i$ to $v_j$ is stronger when they are closer and $v_j$ aligns more with $\psi_i$.

Edges in $\mathcal{E}$ represent possible component transitions. An edge $(v_i, v_j) \in \mathcal{E}$ exists only if:

- $\cos(\varphi_{ij}) > 0$, where $\varphi_{ij}$ is the angle between $\psi_i$ and $\mu_j - \mu_i$, ensuring $\psi_i$ points at least partly toward component $j$, and
- the Bhattacharyya Coefficient [10] between Gaussians of $i$ and $j$ exceeds the user-defined threshold $\eta_{BC}$, ensuring sufficient data connectivity between components.

The edge weight is defined as

$$W((v_i, v_j)) = \frac{||\mu_i - \mu_j||^{\eta_{dist}}}{\cos(\varphi_{i,j})^{\eta_{dir}}}, \tag{3}$$

where $\eta_{dist}$ and $\eta_{dir}$ are user-defined parameters.

The user-defined parameters $(\eta_{BC}, \eta_{dist}, \eta_{dir})$ collectively govern the GG's connectivity and traversal cost, balancing adherence to demonstrations with the flexibility to synthesize novel behaviors. The threshold $\eta_{BC}$ acts as a structural safety filter by dictating the minimum data support required for a transition. Higher $\eta_{BC}$ values strictly enforce empirical support, while lower values suit applications more tolerant of out-of-data movements; however, excessively high thresholds risk fragmenting the GG. For viable edges, $\eta_{dist}$ and $\eta_{dir}$ act as cost regularizers on spatial separation and dynamic consistency, respectively. A high $\eta_{dist}$ heavily penalizes long jumps, favoring paths of closely packed Gaussians, while a high $\eta_{dir}$ enforces strict directional alignment between a Gaussian's inherent dynamics and the physical path to the next vertex. Conversely, lowering these two parameters widens the transition "field-of-view", enabling the graph to bridge disjoint tasks at the risk of leaving data-supported regions.

Depending on the application, some demonstrations may be safely executed in reverse. We refer to this as the *bi-directionality assumption*. The graph is then expanded by creating a counterpart vertex $v_i'$ for every original vertex $v_i$ corresponding to such a demonstration, with reversed dynamics $\mathbf{A}_i' = -\mathbf{A}_i$ and direction $\psi_i' = -\psi_i$.

Finally, the methods in subsequent sections will primarily use shortest paths in the GG. For this reason, the GG can be reduced in size by removing every edge $(v_i, v_j) \in \mathcal{E}$ not included in the shortest path from $v_i$ to $v_j$. This operation does not have any effect on shortest paths in the GG since such edges would never be included, however, it does improve computation times.

## IV. Demonstration Stitching

In this section, we present four methods for stitching together separate demonstrations into a single time-invariant motion policy. These four methods arise from the combination of two ways to use the GG (*Shortest Path* and *Shortest Path Tree*) with two ways of reusing elements from the initial LPV-DS policies $\{f_{\mathcal{D}} \mid \mathcal{D} \in \mathbb{D}\}$.

### A. Graph Strategies

*1) Shortest Path:* We assume for this method that an initial position $\xi_0$ and attractor $\xi^*$ are given. A vertex is added to the GG for each $v_0$ and $v^*$ respectively, with $\mu_0 = \xi_0$ and $\mu^* = \xi^*$. For $v^*$, only incoming edges connect it to existing vertices; for every $v_j \in \mathcal{V}$, (3) is used to compute an initial weight $w_{j,*}$, with the same requirement that $\cos(\varphi_{j,*}) > 0$. However, the weight is further modified to account for data support provided by $v_j$, giving $W((v_j, v^*)) = w_{j,*}/\mathcal{N}(\xi_0 \mid \mu_j, \mathbf{\Sigma}_j)$. On the other hand, only outgoing edges from $v_0$ are added. Since $v_0$ has no notion of direction, for every $v_j \in \mathcal{V}$, $v_0$ adopts direction $\psi_j$ when calculating $w_{0,j}$ with (3), also requiring $\cos(\varphi_{0,j}) > 0$. Factoring in the amount of data-support provided by $v_j$, we get $W((v_0, v_j)) = w_{0,j}/\mathcal{N}(\xi_0 \mid \mu_j, \mathbf{\Sigma}_j)$. A standard shortest-path algorithm (e.g., Dijkstra's [11]) is used to find the shortest path $\langle v_0, v_1, \ldots, v_e, v^* \rangle$ from $v_0$ to $v^*$, with the set $\sigma = \{v_1, \ldots, v_e\} \subseteq \mathcal{V}$ containing the vertices corresponding to components.

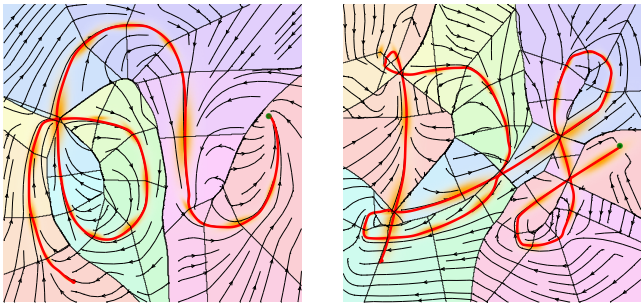
*2) Shortest Path Tree:* Given only an attractor $\xi^*$, a corresponding vertex $v^*$ is added to the GG in the same way as above. The shortest path tree to $v^*$ is found, with all vertices in $\mathcal{V}$ for which there exists a path to $v^*$ stored in $\sigma$. This approach yields a global policy that is agnostic to a starting position. The bi-directionality assumption, however, introduces a critical problem: a path to $v^*$ may exist from both a vertex $v_i$ and its reverse counterpart $v_i'$. Including both vertices in the final policy is problematic, as their opposing dynamics ($\psi_i = -\psi_i'$) effectively cancel each other out. To resolve this ambiguity, we prune the vertex set by applying a simple heuristic: for any such pair $\langle v_i, v_i' \rangle$, we retain only the vertex with the shorter path to $v^*$.

### B. Levels of DS Reuse

Using the set of vertices $\sigma$ from either of the two graph strategies introduced above, a new motion policy can be constructed. Intuitively, reusing components from the initial policies $\{f_{\mathcal{D}} \mid \mathcal{D} \in \mathbb{D}\}$ simplifies computation — but also constrains the new policy. We consider two levels of reuse:

*1) No Reuse:* When LPV-DS is applied to a demonstration $\mathcal{D} \in \mathbb{D}$, data points in $\mathcal{D}$ are clustered to each of the $K_{\mathcal{D}}$ Gaussian components, resulting in the clustering $C_{k,\mathcal{D}} \subseteq \mathcal{D}$ for each $k = 1, \ldots, K_{\mathcal{D}}$. In this method, all data points clustered to vertices in $\sigma$ are collected, $\bigcup_{n_i \in \sigma} C_i$, and a completely new LPV-DS model is fitted. Essentially, the GG is used to filter data points from the demonstrations, after which LPV-DS is applied anew. With the bi-directionality assumption, the data points' velocity $\dot{\xi}$ associated with a reversed vertex are negated (that is, $-\dot{\xi}$ is used).

*2) Reusing Gaussians:* This method reuses the GMM parameters $\theta_i^{GMM} = \{\pi_i, \mu_i, \mathbf{\Sigma}_i\}$ and corresponding data points $C_i$ for each vertex $v_i \in \sigma$. The priors $\{\pi_i\}$, however, are normalized to sum to 1. With the GMM structure fixed, only the second LPV-DS step is performed to find the DS parameters $\theta_i^{DS} = \mathbf{A}_i$ and a shared Lyapunov parameter $\mathbf{P}$ that best fits the filtered data points.



(a) (b)

Fig. 2: Examples of two DS-Chains, with the resulting motions when starting at an initial position in red. Colored regions show the active DS at the corresponding part of the trajectory, with the underlying Gaussians shown in orange.

We explored not only reusing Gaussians but also $\mathbf{P}$ to only recompute the dynamics $\mathbf{A}_i$ or reusing $\mathbf{A}_i$, and instead seek a single $\mathbf{P}$. However, those approaches yielded relatively low performance and are therefore omitted.

## V. Demonstration Chaining

The overall approach of *Demonstration Chaining* is to transition through a sequence of simpler DSs to collectively realize motions more sophisticated than those achievable with any single time-invariant DS alone. For instance, the motions shown in Fig. 2 are impossible for standard LPV-DS or the stitching methods presented in the previous section: they require multiple velocities at a single position — violating time-invariance — and follow paths incompatible with the decay of any parameterized Lyapunov function. In this section, we define a *DS-Chain*, formulate criteria under which it is GAS with respect to an attractor $\xi^*$, and then describe our specific method for using the GG to construct such a GAS DS-Chain.

*Definition 5.1 (DS-Chain):* A DS-Chain is a tuple $\mathcal{C} = \langle \mathcal{F}, \Gamma, \mathcal{T}, \mathcal{U} \rangle$, where $\mathcal{F} = \langle f_1, \ldots, f_N \rangle$ contains $N$ dynamical systems; $\Gamma = \langle \gamma_1, \ldots, \gamma_{N-1} \rangle$ and $\mathcal{T} = \langle \tau_1, \ldots, \tau_{N-1} \rangle$ contain $N-1$ *trigger* and *timer* predicates, respectively, each mapping to $\{\top, \bot\}$; and $\mathcal{U} = \langle u_1, \ldots, u_{N-1} \rangle$ contains $N-1$ dynamical systems.

We model a DS-Chain $\mathcal{C}$ as a finite hybrid automaton with *nominal states* $S = \{s_1, \ldots, s_N\}$ and *intermediate states* $S' = \{s'_1, \ldots, s'_{N-1}\}$. The system initializes in $s_1$. Each nominal state $s_i$ transitions to $s'_i$ when the trigger $\gamma_i$ is $\top$, and each intermediate state $s'_i$ transitions to $s_{i+1}$ when the timer $\tau_i$ is $\top$. Thus, the automaton's structure is $s_1 \xrightarrow{\gamma_1} s'_1 \xrightarrow{\tau_1} s_2 \xrightarrow{\gamma_2} s'_2 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_{N-1}} s_N$. The system dynamics are defined by the state of the DS-Chain: when in state $s_i$ the dynamics are governed by $f_i$, when in state $s'_i$ the dynamics are governed by $u_i$. Fig. 2 visualizes this, with colored regions corresponding to the active nominal state in $S$ and associated DS in $\mathcal{F}$ during that part of the trajectory. Note that the transition DSs in $\mathcal{U}$ are not visualized. Theorem 5.1 provides criteria under which $\mathcal{C}$ is GAS with respect to $\xi^*$.

*Theorem 5.1 (Global Asymptotically Stable DS-Chain):* A DS-Chain $\mathcal{C} = \langle \mathcal{F}, \Gamma, \mathcal{T}, \mathcal{U} \rangle$ — modeled by states in $S$ and $S'$ — is GAS with respect to an attractor $\xi^*$ if:

1) the DSs in $\mathcal{F}$ and $\mathcal{U}$ are bounded,
2) $\forall i = 1, \ldots, N-1$, trigger $\gamma_i$ is guaranteed to evaluate to $\top$ within a finite time when in state $s_i$,
3) $\forall i = 1, \ldots, N-1$, $\tau_i$ is guaranteed to evaluate to $\top$ within a finite time when in state $s'_i$,
4) $f_N$ is GAS with respect to $\xi^*$.

*Proof:* Since $\gamma_i$ is guaranteed to evaluate to $\top$ within a finite time when in state $s_i$, it holds that the transition from $s_i$ to $s'_i$ will occur within a finite time. Likewise, since $\tau_i$ is guaranteed to evaluate to $\top$ within a finite time when in state $s'_i$, it holds that the transition from $s'_i$ to $s_{i+1}$ will occur within a finite time. Thus, the system initializes in state $s_1$ and is guaranteed to transition through all states to eventually arrive and remain in $s_N$. Since all dynamics $f_1, \ldots, f_N$ and $u_1, \ldots, u_{N-1}$ are bounded, the system's state will be bounded when entering $s_N$, from which the dynamics are governed by $f_N$ — with $f_N$ being GAS with respect to $\xi^*$. In other words, starting from any bounded position and following the dynamics specified by $\mathcal{C}$, eventually the system's state will be bounded and the dynamics will be governed by $f_N$, inevitably leading to $\xi^*$. Therefore, $\mathcal{C}$ is GAS with respect to $\xi^*$. ■

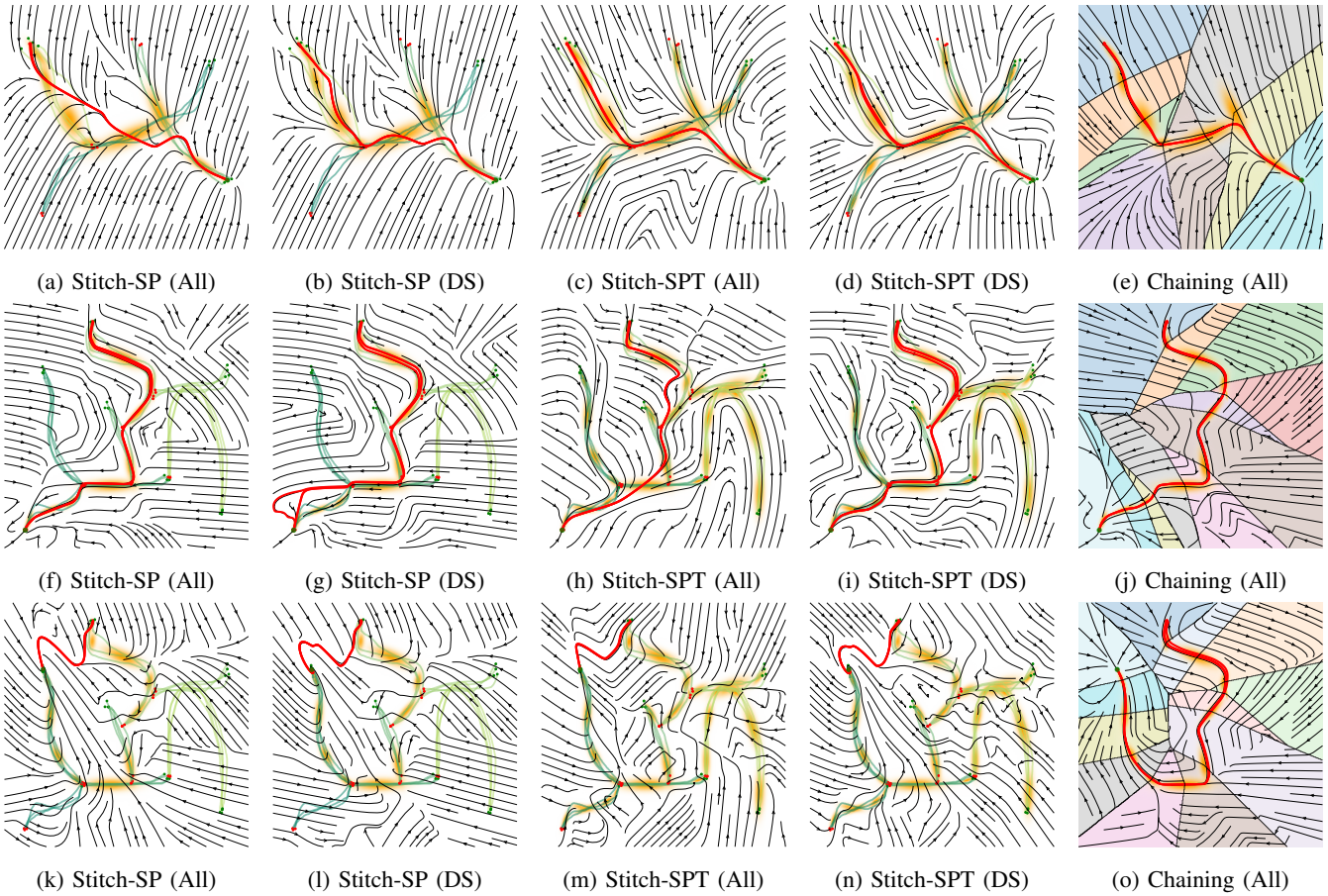
### A. Constructing a GAS DS-Chain using the Gaussian Graph

We now define our specific method of creating a DS-Chain using the GG, showing that it satisfies every criterion of Theorem 5.1 and is therefore GAS.

*1) Dynamical systems in $\mathcal{F}$:* From the Gaussian graph $\mathcal{G}$, a sequence of vertices $\sigma = \langle v_1, v_2, \ldots, v_M \rangle$ can be obtained by, for instance, finding the shortest path from an initial position $\xi_0$ to $\xi^*$. Each pair $(v_i, v_{i+1})$ is an edge in $\mathcal{E}$. Just as in Section IV-A.1, we do not include the vertices $v_0$ and $v^*$ corresponding to $\xi_0$ to $\xi^*$ in $\sigma$.

Our approach is to partition $\sigma$ using a sliding window to fit a local DS to each segment. While the simplest approach uses vertex pairs (i.e., single edges), we achieve better results using overlapping vertex triplets. Triples are advantageous because adjacent segments share exactly one edge, providing a continuous transition region between successive local models. This yields a sequence of $M-2$ overlapping segments: $\langle \langle v_1, v_2, v_3 \rangle, \langle v_2, v_3, v_4 \rangle, \ldots, \langle v_{M-2}, v_{M-1}, v_M \rangle \rangle$. We make one assumption on the path $\sigma$: no segment derived from $\sigma$ contains any vertex *and* its reversal (under the bi-directionality assumption) due to the difficulty of finding a DS from the resulting set of data points.

LPV-DS is applied to each segment $\langle v_i, v_{i+1}, v_{i+2} \rangle$ to obtain an LTI DS $f_i$ that is GAS at the final vertex's position, $\mu_{i+2}$. While it seems intuitive to fit $f_i$ to all data points clustered to these three vertices, we find that LPV-DS more reliably obtains a DS when omitting a specific subset of data around the target vertex $v_{i+2}$. Specifically, let $l = ||\mu_{i+2} - \mu_{i+1}||_2$ be the distance between the last two vertices. To improve fitting, we discard any reference point $\xi^{ref}$ belonging to $v_{i+2}$ that is both farther from $v_{i+1}$ than $l$



(a) Stitch-SP (All) (b) Stitch-SP (DS) (c) Stitch-SPT (All) (d) Stitch-SPT (DS) (e) Chaining (All)

(f) Stitch-SP (All) (g) Stitch-SP (DS) (h) Stitch-SPT (All) (i) Stitch-SPT (DS) (j) Chaining (All)

(k) Stitch-SP (All) (l) Stitch-SP (DS) (m) Stitch-SPT (All) (n) Stitch-SPT (DS) (o) Chaining (All)

Fig. 3: Qualitative results for datasets `2D Small` (first row) and `2D Large` (second and third row). Simulations are shown in red, demonstrations in blue-green, and Gaussians extracted from the GG in orange. Baseline methods are excluded as they failed in all the presented cases, as well as Chaining (DS) since results are similar to Chaining (All).

(i.e., $||\xi^{ref} - \mu_{i+1}||_2 > l$) and closer to $v_{i+2}$ than $0.1l$ (i.e., $||\xi^{ref} - \mu_{i+2}||_2 < 0.1l$). Observe that $f_i$ is invariant to $\xi_0$ and $\xi^*$ and can therefore be pre-computed and stored in a lookup table (bounded by $|\mathcal{E}| \max_{v \in \mathcal{V}} |\text{neighbors}(v)|$ entries).

A final DS $f^*$ is fitted to data from the last two vertices $v_{M-1}$ and $v_M$, and ensured GAS with respect to the attractor $\xi^*$. This results in $\mathcal{F} = \langle f_1, \ldots, f_{M-2}, f^* \rangle$. As an optional step, an initial DS $f_0$ — GAS with respect to $\mu_1$ and fitted to data from $v_1$ — can be prepended to $\mathcal{F}$. The same levels of reuse as described in Section IV-B also apply here.

*2) Triggers in $\Gamma$:* We define a trigger $\gamma_i(\xi)$ (associated with a segment $\langle v_i, v_{i+1}, v_{i+2} \rangle$) as $\top$ when $\xi$ has passed the second vertex $v_{i+1}$ on its way from $v_i$ to $v_{i+2}$. That is, $\gamma_i(\xi) = \top$ when

$$\frac{||\xi - \mu_i||_2}{||\xi - \mu_{i+2}||_2} \geq \frac{||\mu_{i+1} - \mu_i||_2}{||\mu_{i+1} - \mu_{i+2}||_2}, \tag{4}$$

$\gamma_i(\xi) = \perp$ otherwise. Specifically, for the optional initial DS $f_0$, in (4) we let $\mu_i = \xi_0$, $\mu_{i+2} = \mu_1$, and $\mu_{i+1} = (\xi_0 + \mu_1)/2$.

*3) Timers in $\mathcal{T}$:* For each segment $i = 1, \ldots, N-1$, the timer $\tau_i(t)$ switches to $\top$ after prescribed transition time $T_i$. Specifically, $\tau_i(t) = \top$ when $t - t_i \geq T_i$, $\tau_i(t) = \perp$ otherwise, where $t$ is the current time and $t_i$ is the time at which the DS-Chain enters state $s_i$. We choose $T_i \in \mathbb{R}^+$ to approximate the time required for the state $\xi$ to travel a fraction $\alpha \in [0, 1]$ from $v_{i+1}$ to $v_{i+2}$:

$$T_i = \alpha \frac{||\mu_{i+2} - \mu_{i+1}||_2}{||f_i(\mu_{i+1}) + f_{i+1}(\mu_{i+1})||_2/2} \tag{5}$$

This can be interpreted as follows: state $s_i$ is entered when trigger $\gamma_i = \top$, which occurs near the middle Gaussian $i+1$ of the segment $\langle v_i, v_{i+1}, v_{i+2} \rangle$. Hence, $T_i$ estimates the time needed to travel a fraction $\alpha$ of the distance from $v_{i+1}$ to $v_{i+2}$ under the average velocity of the current and next DS. In particular, $\alpha = 0$ yields an immediate transition, whereas $\alpha = 1$ delays the transition until approximately the full segment has been traversed ($\xi \approx \mu_{i+2}$).

*4) Intermediate dynamics in $\mathcal{U}$:* The system remains in intermediate state $s_i'$ for $T_i$ time while transitioning from state $s_i$ with dynamics $f_i$ to state $s_{i+1}$ with dynamics $f_{i+1}$. Therefore, we let the dynamics $u_i$ during this transition be a linear interpolation between the two DSs:

$$u_i(\xi, t) = \frac{t - t_i}{T_i} f_i(\xi) + \left(1 - \frac{t - t_i}{T_i}\right) f_{i+1}(\xi). \tag{6}$$

Since $f_1, \ldots, f_N$ are all bounded, so too are the linear interpolations $u_1, \ldots, u_{N-1}$.

| Method | 2D Small | | | | 2D Large | | | | 3D PC-GMM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Success % | RMSE | Data Support | Comp. Time | Success % | RMSE | Data Support | Comp. Time | Success % | RMSE | Data Support | Comp. Time |
| Baseline (All) | 20.8% | 1.77 ± 0.72 | 0.73 ± 0.17 | 8.29 ± 0.63 | 19.1% | 2.15 ± 0.74 | 0.72 ± 0.23 | 17.51 ± 1.86 | 28.3% | 0.28 ± 0.33 | 0.88 ± 0.17 | 17.78 ± 16.59 |
| Baseline (DS) | 20.0% | 2.97 ± 1.57 | 0.68 ± 0.25 | 10.16 ± 0.42 | 13.7% | 3.30 ± 0.76 | 0.58 ± 0.24 | 33.65 ± 2.67 | 35.8% | 2.53 ± 1.19 | 0.93 ± 0.10 | 251 ± 23 |
| Stitch-SP (All) | 98.3% | **0.08 ± 0.02** | 0.86 ± 0.24 | 0.76 ± 0.42 | 90.2% | 0.14 ± 0.30 | 0.90 ± 0.17 | 1.09 ± 1.28 | 90.0% | **0.04 ± 0.03** | 0.87 ± 0.19 | 1.33 ± 1.27 |
| Stitch-SP (DS) | 94.2% | **0.08 ± 0.02** | 0.86 ± 0.24 | 0.14 ± 0.71 | 91.3% | 0.19 ± 0.41 | 0.87 ± 0.21 | 0.59 ± 1.90 | 85.8% | **0.04 ± 0.02** | 0.85 ± 0.20 | 0.06 ± 0.04 |
| Stitch-SPT (All) | 97.5% | 0.09 ± 0.03 | 0.86 ± 0.19 | 1.44 ± 0.72 | 89.7% | 0.27 ± 0.44 | 0.77 ± 0.22 | 5.06 ± 5.95 | 81.7% | 0.08 ± 0.11 | 0.92 ± 0.13 | 7.10 ± 12.37 |
| Stitch-SPT (DS) | 95.8% | **0.08 ± 0.02** | 0.85 ± 0.20 | 0.08 ± 0.02 | 91.6% | 0.67 ± 0.82 | 0.80 ± 0.22 | 15.28 ± 16.46 | **100.0%** | 0.45 ± 0.57 | 0.95 ± 0.06 | 196 ± 100 |
| Chaining (All) | **99.2%** | 0.11 ± 0.14* | **0.91 ± 0.14** | 0.26 ± 0.05 | **97.4%** | **0.12 ± 0.18*** | **0.96 ± 0.09** | 0.29 ± 0.09 | 97.5% | 0.05 ± 0.03* | 0.97 ± 0.06 | 0.46 ± 0.32 |
| Chaining (DS) | 97.5% | 0.14 ± 0.19* | **0.91 ± 0.17** | **0.01 ± 0.01** | 94.1% | 0.15 ± 0.39* | 0.95 ± 0.09 | **0.01 ± 0.00** | **100.0%** | 0.06 ± 0.02* | **0.98 ± 0.03** | **0.02 ± 0.00** |

TABLE I: The results on each dataset for the methods LPV-DS (Baseline), Stitching using the Shortest Path (Stitch SP) and Shortest Path Tree (Stitch SPT), and Chaining, showing the success rate in percent, RMSE, Data Support, and computation time in seconds from four random seeds. The two levels of reuse from Section IV-B — recomputing both the GMM and DS parameters (all) or only recomputing the DS parameters (DS) — are tested. *Calculating the RMSE of *Chaining* methods is approximated by mixing DSs equally during transitions.

By construction, $\mathcal{C} = \langle \mathcal{F}, \Gamma, \mathcal{T}, \mathcal{U} \rangle$ satisfies all criteria of Theorem 5.1 and is therefore GAS with respect to $\xi^*$. Criteria 1 and 4 hold because all primary and intermediate dynamics in $\mathcal{F}$ and $\mathcal{U}$ are bounded, and the final system $f_N = f^*$ is GAS. Furthermore, Criteria 2 and 3 are satisfied because every trigger $\gamma_i(\xi)$ evaluates to $\top$ in finite time (since each $f_i$ is GAS towards $\mu_{i+2}$) and the transition timers $\tau_i(t)$ rely on finite $T_i$.

## VI. Results

### A. Simulation Experiments

We evaluate the proposed methods alongside LPV-DS as a baseline on three datasets: `2D Small` (Fig. 1 and Figs. 3a–3e); a larger `2D Large` of 6 demonstrations (Figs. 3f–3o); and `3D PC-GMM` based on the dataset from [9], comprising "3D C-shape top", "3D viapoint 1", and "3D viapoint 2". For each dataset, all pairwise combinations of pooled initial-attractor points (assuming bi-directionality) yield 30 instances for `2D Small`, 182 for `2D Large`, and 30 for `3D PC-GMM`. All experiments use $\eta_{BC} = 0.05$, $\eta_{dist} = 2$ and $\eta_{dir} = 1$, thus slightly prioritizing closeness over alignment, and are rerun using four random seeds.

Results are presented in Table I. The *success-rate* is shown, with a success meaning that a valid solution was found and the goal state was reached within 1000 simulation seconds. *Root Mean-Squared Error* (RMSE) is taken between $\dot{\xi} = f(\xi^{ref})$ and $\dot{\xi}^{ref}$. Due to these methods aggregating reference points from different parts of various demonstrations, we found the common *Dynamic Time Warping Distance* to be an unsuitable measurement of how data-supported a trajectory is. Instead, a trajectory's *Data Support* is the average evaluation of its points. A point with distance $d$ to the nearest reference point evaluates to 1 if $d < \mu_{\mathbb{D}}$, else $\exp\left(-\frac{1}{2}\left(\frac{d-\mu_{\mathbb{D}}}{\sigma_{\mathbb{D}}}\right)^2\right)$, where $\mu_{\mathbb{D}}$ and $\sigma_{\mathbb{D}}$ are the mean and standard deviation of inter-trajectory nearest-neighbor distances over reference points in demonstrations in $\mathbb{D}$. Thus, a Data Support of 1.0 indicates the trajectory to be as close to the data as the reference trajectories, while 0.0 indicates no support. Metrics are averaged over successful solutions. More details are available on our repository[1].

As shown in Table I, our proposed GG methods successfully synthesize new trajectories, achieving substantially higher success rates across all datasets compared to the LPV-DS baselines, which fail in the majority of cases. Furthermore, in the instances where valid solutions are found, our approaches generally yield higher-quality trajectories. This is evidenced by improved alignment with reference velocities (lower RMSE) and stronger adherence to demonstrated regions (higher Data Support) compared to baseline methods. Our proposed methods additionally yield lower inference times, with *Chaining (DS)* outperforming the others by a significant margin. This rapid online performance is achieved by shifting much of the computational load to offline precomputations, which averages 4.2 seconds for *Chaining (DS)* and 44.3 seconds for *Chaining (All)*.

Fig. 3 illustrates qualitative examples of our proposed methods. Notably, the structural difference between the *Stitch-SP* and *Stitch-SPT* methods is visible: *Stitch-SP* yields a DS tailored to a specific initial position, whereas *Stitch-SPT* remains agnostic to the initial position and thereby generates a global DS toward the target. The bottom row (Figs. 3k–3o) highlights the particular strengths of *Chaining* in complex scenarios where *Stitching* fails. Specifically, *Chaining* is able to formulate and execute sophisticated trajectories that are otherwise challenging for a single time-invariant DS to represent while satisfying the strict decay constraints of a Lyapunov function.

Overall, the simulation results confirm that both *Stitching* and *Chaining* effectively enable zero-shot generalization to unseen tasks with a relatively high rate of success.

### B. Real-Robot Experiments

Furthermore, we validate our simulated findings through real-robot experiments using the two methods with the highest success rate on `3D PC-GMM`: *Stitch-SPT (DS)* and *Chaining (DS)*. The experiments are conducted on a UR3e manipulator equipped with an OnRobot two-finger gripper, using the same configurations as in the previous section. We collect training data for three distinct demonstrations (two trajectories each): (i) grasping object 1 starting from box A, (ii) grasping object 2, and (iii) transitioning from above object 1 to box B (see first row of Fig. 4). Importantly, grasping solely refers to moving towards the object and closing the gripper without lifting it.

Using this dataset, we evaluate zero-shot generalization on two unseen composite tasks, both starting from the workspace center: (1) picking up object 2 and moving it




Fig. 4: Real-robot experiments where we collect 3 demonstrations with 2 trajectories each and test *Stitch-SPT (DS)* and *Chaining (DS)* on two different tasks that require reversing and stitching. The environment consists of a transparent cube (object 1), a red cube (object 2), a brown box (box A), a white box (box B), and a plant acting as an obstacle.

to box B, and (2) picking object 1 and moving it to box A. In order to solve task (1), the underlying policy needs to combine all the existing demonstrations. Conversely, task (2) requires the system to reverse demonstration (i) without experiencing interference from other data. As the DS outputs end-effector velocities, we map them to joint velocities via an inverse-kinematics solver and execute them with a velocity controller. Gripper actions are triggered when reaching the desired pickup location. Additionally, a complete task is defined by consecutive attractor positions. For instance, in task (1), the attractor is initially object 2, but then changed to box B once object 2 is grasped.

Fig. 4 illustrates the resulting behaviors for both *Stitch-SPT (DS)* and *Chaining (DS)*. As can be seen, both methods successfully complete the tasks, confirming the efficacy of our methods as observed in the simulation datasets. Crucially, integrating trajectory stitching (or chaining) combined with the bi-directionality assumption enables the end-effector to freely move around the whole workspace while remaining within the demonstrated data manifold, allowing it to closely follow the most fitting demonstration at any given time.

## VII. Discussions

### A. Methodological Trade-Offs

Each of the proposed methods offers distinct advantages while presenting specific trade-offs. *Stitch-SP* constructs a DS tailored to a specific initial-target pair; it generally achieves trajectory qualities comparable to *Stitch-SPT*, but requires less computation time. On the other hand, *Stitch-SPT* generates a global DS dependent exclusively on the target position, resulting in a policy that generally accommodates any initial position within data support and thereby making it inherently more robust to large perturbations. Furthermore, it can be entirely precomputed if the set of desired target positions is known in advance.

*Chaining* yields the highest-quality trajectories and extremely fast computation times, provided the offline precomputation is completed. Crucially, *Chaining* is the only method among those evaluated capable of producing highly sophisticated, intersecting trajectories (such as those in Fig. 2 and Figs. 3k–3o) that exceed the representational capacity of a single time-invariant DS. However, because *Chaining* formulates an inherently time-varying DS, it relies on the implicit assumption that the system state remains relatively close to the active graph segment. This results in a higher sensitivity to large perturbations compared to the *Stitching* approaches. Ultimately, the optimal choice of method is application-dependent, balancing the need for robustness, computational performance, and trajectory complexity.

### B. Incremental Learning and Scalability

One of the most significant advantages of the proposed methods is their scalability to unseen tasks within a shared workspace. Rather than collecting a new demonstration for

every possible task, existing demonstrations can be reused to synthesize novel trajectories. If an entirely new region of the workspace must be reached, only a brief, localized demonstration connecting the new area to the existing spatial network is required. Thus, our methods integrate naturally with incremental learning. Incorporating a new demonstration requires fitting an LPV-DS model to the new demonstration, translating it into new vertices and edges, and appending them to the existing GG. Once updated, *Stitching* or *Chaining* can be applied. Notably, prior *Chaining* precomputations can be retained; the system only needs to compute and store the localized dynamics of newly added segments. This flexibility is key for deployment in real-life settings where adjustability and extendibility are critical.

### C. Limitations and Future Work

While the proposed framework successfully enables zero-shot generalization, it presents a few limitations — beyond those already discussed — that motivate future work. First, the construction of the GG currently relies on user-defined hyperparameters $(\eta_{BC}, \eta_{dist}, \eta_{dir})$ to balance data-overlap, spatial distance, and dynamic consistency. Besides exploring the wide range of alternative edge-weight formulations, future work could aim to automate the parameter selection process. This could be achieved by framing the graph connectivity as an optimization problem, or by learning acceptable parameter bounds directly from the demonstration data distribution.

Second, our approach aims to synthesize novel trajectories within data support. However, it does not explicitly account for changing object positions or obstacles introduced into the workspace after the demonstrations were collected. Integrating real-time obstacle avoidance — for instance, using DS *modulation* [12] or *Control Barrier Functions* [13] — could be implemented both during the GG path-finding phase and within the resulting control policy. Similarly, combining our method with, for instance, *Elastic Motion Policies* [6] would allow us to adapt to different object positions.

Finally, the current implementation operates primarily on spatial components (Euclidean space). A highly promising avenue for future work is extending the approach to include end-effector orientations by integrating SE(3) LPV-DS [5]. The primary modification required would be adapting the edge-weight metrics used during the graph search to account for rotational manifolds. This extension highlights the underlying generality of the Gaussian Graph framework and provides a strong theoretical foundation for scaling to more complex robotics tasks.

## VIII. Conclusions

Although there are multiple ways to learn motion policies from expert demonstrations, dynamical system (DS) based methods have proven to be data-efficient and provide fast and reactive control with stability guarantees. However, these methods often lack the ability to combine learned skills to generalize to new scenarios. In this work, we presented a framework to effectively perform zero-shot generalization from motion demonstrations of separate tasks in a shared workspace to new, undemonstrated tasks. To this end, we introduced the *Gaussian Graph* (GG), constructed from the components obtained by applying Linear Parameter-Varying DS (LPV-DS) to each demonstration separately. The GG thereby allows for the unification of continuous control with well-known graph algorithms. We then introduced two classes of methods: *Demonstration Stitching*, which combines GG components along a shortest path or in a shortest path tree to produce a time-invariant DS; and *Demonstration Chaining*, which divides a path in the GG into "stepping stones" that are sequentially traversed. The Stitching methods naturally inherit LPV-DS's stability guarantees while reliably generalizing to new tasks, while for Chaining we provide formal proofs of stability and showcase the flexibility it provides in the resulting types of motions. Simulations and real-robot evaluations show their efficacy for zero-shot generalization where baselines fail. Overall, this work contributes towards developing methods that are sample-efficient, reactive, and provably convergent while effectively reusing data for unseen tasks.

## References


[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *Advanced Robotics*, vol. 38, no. 18, pp. 1232–1254, 2024.

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

[5] S. Sun and 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)*, pp. 5152–5159, IEEE, 2024.

[6] T. Li, S. Sun, S. S. Aditya, and N. Figueroa, "Elastic motion policy: An adaptive dynamical system for robust and efficient one-shot imitation learning," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9846–9853, 2025.

[7] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, pp. 1–29, Jan 2016.

[8] J. R. Medina and A. Billard, "Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models," in *Conference on Robot Learning*, pp. 175–184, PMLR, 2017.

[9] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Conference on Robot Learning*, pp. 927–946, PMLR, 2018.

[10] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distribution," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–110, 1943.

[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: his life, work, and legacy*, pp. 287–290, 2022.

[12] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, pp. 433–454, May 2012.

[13] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, 2019.