IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

An Alignment-Based Approach to Learning Motions From Demonstrations

Alex Cuellar, *Graduate Student Member, IEEE*, Christopher K. Fourie, and Julie A. Shah

***Abstract*—Learning from Demonstration (LfD) has shown to provide robots with fundamental motion skills for a variety of domains. Various branches of LfD research (e.g., learned dynamical systems and movement primitives) can generally be classified into "time-dependent" or "time-independent" systems. Each provides fundamental benefits and drawbacks – time-independent methods cannot learn overlapping trajectories, while time-dependence can result in undesirable behavior under perturbation. This letter introduces Cluster Alignment for Learned Motions (CALM), an LfD framework dependent upon an alignment with a representative "mean" trajectory of demonstrated motions rather than pure time- or state-dependence. We discuss the convergence properties of CALM, introduce an alignment technique able to handle the shifts in alignment possible under perturbation, and utilize demonstration clustering to generate multi-modal behavior. We show how CALM mitigates the drawbacks of time-dependent and time-independent techniques on 2D datasets and implement our system on a 7-DoF robot learning tasks in three domains.**

***Index Terms*—Learning from demonstration, probabilistic inference, physical human-robot interaction.**

I. Introduction

As robots are introduced in industry and domestic settings, there is increasing need for robots to learn fundamental motions for given tasks. Learning from demonstrations (LfD) provides operators the ability to teach robots motions, ideally from a handful of demonstrations (∼2 to 5) [1]. Such work allows robots to learn specific actions combining to complete more complex tasks. Existing methods can be partitioned into time-independent approaches, such as learned dynamical systems [2], [3], and time-dependent approaches, such as movement primitives [4], [5]. Despite advances in both areas, core limitations remain: time-independent methods cannot model overlapping trajectories, while time-dependent methods can exhibit undesirable behavior under perturbation – if a robot is moved to another region of demonstrations, such methods have difficulty picking up where a perturbation leaves off [1].

We introduce CALM (Cluster Alignment for Learned Motions), a novel LfD approach dependent on *alignment* – a mapping of states in the robot trajectory to those in one or more representative "mean" trajectories for different clusters of demonstrations. CALM determines which mean trajectory to follow, and consistently updates its alignment to determine which section of the chosen mean trajectory to follow. Additionally, if for any reason (e.g., a perturbation) the robot's partial trajectory aligns better to another cluster's mean trajectory, CALM will dynamically align to this new cluster and follow it (see Fig. 1). By consistently updating an alignment between the robot and demonstrated trajectories, CALM mitigates the limitations of time-dependent and -independent methods.

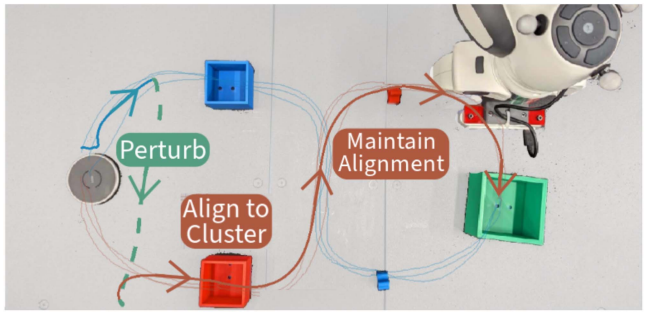

Fig. 1. Example of CALM's learned behavior. There are two demonstration clusters. One cluster (*blue*) passes the blue box and marker, while the other (*red*) passes the red box and marker. The robot starts with *blue*, but is perturbed and aligns to *red*. CALM maintains alignment despite cluster overlap.

While this letter introduces, to our knowledge, the first approach to LfD that uses partial trajectory alignment, techniques for trajectory alignment are not new [6], [7]. However, existing methods generally assume (a) initial points of two trajectories align and (b) no sudden and large jumps in alignment. While reasonable in many settings, our need to maintain alignment during perturbations necessitates relaxing these constraints.

The primary contributions of this letter are as follows:

1) An alignment-dependent controller (CALM) for learning and reconstructing demonstrated motions.
2) An HMM-based alignment method capable of handling alignment discontinuities from perturbation.
3) A method for choosing which of multiple demonstration clusters CALM follows given an alignment.

We also compare the CALM framework to time-dependent and time-independent approaches on various 2D datasets, and demonstrate CALM's viability in multiple robot domains.[1]

II. Related Works

For ease of training, CALM is designed to learn motions from limited demonstrations (∼2-5). Therefore, we distinguish

Received 11 February 2025; accepted 30 August 2025. Date of publication 24 September 2025; date of current version 13 October 2025. This article was recommended for publication by Associate Editor K. Gopalakrishnan and Editor A. Faust upon evaluation of the reviewers' comments. This work was supported in part by NSF grant on AI Coaching under Grant IIS-2204914 and in part by Los Alamos National Laboratories under Grant C3464. *(Corresponding author: Alex Cuellar.)*

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: alexcuel@mit.edu; ckfourie@csail.mit.edu; julie_a_shah@csail.mit.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3613956, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3613956

[1]Implementation and comparisons: https://github.com/AlexCuellar/CALM

2377-3766 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

©2026 IEEE

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

CALM from recent advances via large learning models such as diffusion [8] and action chunking transformers [9]. Such methods show impressive capabilities in policy learning; however, even modern approaches require many demonstrations [10]. Instead, we compare CALM to movement primitives, learned DS systems, and other lightweight methods effective when learned motions can be deployed more predictably. These methods can be partitioned into time-dependent and time-independent approaches, each subject to inherent limitations.

1) Time-Independent LfD: Time-independent LfD takes the form of learned dynamical systems (DS) that map robot state $\mathbf{x}$ to desired velocity $\dot{\mathbf{x}}$. Many approaches are based in Khansari-Zadeh et al.'s SEDS framework, which guarantees stability to a goal state [2], [11], while others learn a DS via neural networks [12], [13]. However, DS formulations fundamentally cannot represent trajectories that overlap with themselves. This is because DS's map each state $\mathbf{x}$ to one heading $\dot{\mathbf{x}}$. In contrast, an overlapping trajectory must map the same state to different headings depending on which trajectory segment is followed.

2) Time-Dependent LfD: Time-dependent LfD methods largely center around Movement Primitives (MPs). MPs are broadly classified into two categories, with extensions for various desired behaviors [14], [15]. First, Dynamic Movement Primitives (DMPs) combine a learned forcing function with a linear attractor, providing stability grantees [4]. However, DMPs only learn from one demonstration, making them inherently uni-modal. Probabilistic Movement Primitives (ProMPs) generate a distribution over trajectories which can be conditioned on a desired initial point or via-points [5]. While probabilistic conditioning can produce multi-modal behavior, the lack of dynamics can lead to physically unrealistic trajectories and discontinuities [15].

Others break from the MP structure entirely. Nawaz et al., for example, uses Neural ODEs to learn a DS (a time-independent formulation) augmented by a time-dependent correction that ensures stability and safety around obstacles [16]. Despite this time-dependent correction, the underlying DS means it has trouble with overlapping trajectories similar to purely time-independent formulations (see Section V-A).

No matter the formulation, all time-dependent methods have potentially undesirable behavior when perturbed: while time-independent methods can adjust to a perturbation's endpoint, time-dependence causes trajectories to "snap" back to the region dictated by the current timestep. To mitigate such behavior, others seek less strict time-dependency. Calilnon et al., for example, introduced a technique using Hidden Semi-Markov Models (HSMM) to determine which of several linear systems to follow [17], [18]. The HSMM depends both on state and time; thus the section of demonstrations followed by the system is informed by (but not strictly dependent on) time. Therefore, if the robot is held in place for some time, it can continue smoothly once released even if time has passed. However, this method is not provably stable, and large perturbations may cause unfamiliar transitions between attractors, confusing the learned transition function.

Via alignment, CALM overcomes key limits: without enforcing one heading per state, it models overlapping motions, and time-independence enables natural perturbation recovery. By clustering demonstrations, CALM also captures multi-modal behavior that is stable about each mode's endpoint.

III. CALM Framework

In contrast to prior work, Cluster Alignment for Learned Motions (CALM) determines a robot's velocity via an 'alignment" between its trajectory and a mean of demonstrations. CALM is globally asymptotically stable in general, and globally asymptotically stable about demonstrations' endpoint (an important quality shared by prior methods [3], [4]) given requirements on our alignment. We use Fourie et al.'s probabilistic TRACER technique to cluster demonstrations and generate a mean trajectory for each cluster [19]. TRACER is an Expectation Maximization algorithm that iteratively determines the probability of each trajectory corresponding to each mean trajectory, and updates mean trajectories to best represent the trajectories in its cluster. Updates of the mean trajectory minimize the Dynamic Time Warping (DTW) cost between each mean and its corresponding trajectories. TRACER can use a fixed number of clusters or infer one, letting operators decide whether to impose structure on demonstrations.

We introduce the problem setting in Section III-A. Then Section III-B describes the control framework and Section III-C describes stability requirements. Finally, in Section III-D, we discuss how CALM produces motions which match the velocity profile of a mean trajectory.

A. Problem Setting

Let $\mathbf{x}^m$ be a mean trajectory of demonstrations and $\mathbf{x}^r$ be the robot's trajectory thus far. We model trajectories as sequences of states, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n]$, each representing end effector coordinates $\mathbf{x}_i \in \mathbb{R}^d$. We notate the velocity at each state as $\dot{\mathbf{x}}_i \in \mathbb{R}^d$. States in $\mathbf{x}^m$ and $\mathbf{x}^r$ represent each trajectory at constant time intervals δ^m and δ^r respectively.

Additionally, let alignment be represented by a "task" variable, $\tau_k \in \mathbb{N}$, which maps state $\mathbf{x}^r_k$ in the robot's trajectory to a state in the mean trajectory $\mathbf{x}^m$. This task variable indicates the robot trajectory's progress relative to the mean trajectory ($\tau_k = i$ indicates that state $\mathbf{x}^m_i$ best describes the robot's progress along the mean trajectory at state $\mathbf{x}^r_k$). Taking inspiration from prior work, we use a probabilistic interpretation of alignment [7], [19]. $P(\tau_k = i|\mathbf{x}^r, \mathbf{x}^m)$ is the probability that $\mathbf{x}^r_k$ aligns to $\mathbf{x}^m_i$, with $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$ being the corresponding distribution over all values of τ_k.

B. CALM Controller

In this letter, we formulate a controller as follows:

$$\dot{\mathbf{x}} = k_v \frac{\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m)}{||\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m)||} \tag{1}$$

$\mathbf{x} \in \mathbb{R}^d$ is the robot state, and k_v controls speed. Function g uses the robot state and history ($\mathbf{x}^r$) to assess alignment with $\mathbf{x}^m$, and following ∇g moves the robot toward better alignment. Normalization in (1) separates direction from speed, ensuring the velocity matches demonstrations. We define g as:

$$g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m) = \sum_{i=1}^{F} q_i(\mathbf{x}) P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m) \tag{2}$$

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Where $\mathbf{x}^r$ currently has k states, $\mathbf{x}^m$ has F states, and $q_i(\mathbf{x}) := p(\mathbf{x}|\tau_k = i, \mathbf{x}^m)$ is a Gaussian centered on $\mathbf{x}_i^r$ (thus, $q_i(\mathbf{x})$ increases when $\mathbf{x}$ is near $\mathbf{x}_i^m$):

$$p(\mathbf{x}|\tau_k = i, \mathbf{x}^m) = \mathcal{N}(\mathbf{x}|\mathbf{x}_i^m, \Sigma_m) \tag{3}$$

Additionally, notice that $P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m)$ is an alignment probability. However, as $\mathbf{x}^r$ has k states, τ_{k+1} represents the alignment between the robot's *next* state and the mean trajectory. To calculate a distribution over the robot trajectory's next alignment, we introduce a static transition probability:

$$\theta_{j\to i} := P(\tau_{k+1} = i|\tau_k = j) \tag{4}$$

Using the assumption $\tau_{k+1} \perp\!\!\!\perp \mathbf{x}^r, \mathbf{x}^m|\tau_k$:

$$P(\tau_{k+1}=i|\mathbf{x}^r, \mathbf{x}^m) = \sum_{j=1}^{F} \theta_{j\to i} P(\tau_k = j|\mathbf{x}^r, \mathbf{x}^m) \tag{5}$$

Under the definition in (2), $g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m)$ is a mixture of Gaussians, and its gradient is as follows:

$$\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m) = \sum_{i=1}^{F} \nabla_{\mathbf{x}} q_i(\mathbf{x}) P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m) \tag{6}$$

$$= \sum_{i=1}^{F} \Sigma_m^{-1}(\mathbf{x}_i^m - \mathbf{x}) q_i(\mathbf{x}) P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m) \tag{7}$$

Here, the gradient of g will push the robot towards the points in $\mathbf{x}^m$ to which it will most likely align next. Consistently updating the robot alignment distribution $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$ results in each gradient step pushing the robot further along demonstrations' mean trajectory $\mathbf{x}^m$.

C. Stability Guarantees

Since g is a mixture of Gaussians weighted by alignment $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$, its gradient is globally asymptotically stable. Furthermore, requirements on the alignment probability can guarantee that the system is globally asymptotically stable about the final point of the mean trajectory $\mathbf{x}_F^m$:

Theorem 1: The CALM system defined in (1) is globally asymptotically stable. *(See proof in Appendix A)*

Theorem 2: If $\lim k \to \infty P(\tau_k = F|\mathbf{x}^r, \mathbf{x}^m) = 1$ and $\theta_{j\to i} = 0 \,\forall\, i < j$, the robot state will converge to the mean trajectory's final point: $\mathbf{x} - \mathbf{x}_F^m \to 0$. *(See proof in Appendix B)*

For this letter, we use the transition function as follows:

$$\theta_{j\to i} \propto \begin{cases} \phi(i, j+\Delta) & i \geq j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Where $\Delta = \delta^r/\delta^m$ is the ratio of time intervals between states in each trajectory and $\phi(\cdot,\cdot)$ is the radial basis function:

$$\phi(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{||\mathbf{a}-\mathbf{b}||^2}{2\sigma}\right) \tag{9}$$

for a given σ. The transition probability is proportional to a radial basis function centered on a point slightly ahead of the current alignment (i.e., $\tau_k + \Delta$). However, each successive point of the robot trajectory cannot align to a prior point in the mean trajectory (i.e., $\theta_{j\to i} = 0$ if $i < j$). This satisfies the transition function requirement in Theorem 2.

D. Velocity Matching

To accurately replicate motions from demonstrations, the velocity gain k_v in (1) must match the speed of motions demonstrated. However, the robot should also ensure fast recovery from perturbations. Therefore, we formulate k_v as a linear combination of an "alignment" gain k_v^a which matches the speed of demonstrations and a "perturbed gain" k_v^p which pulls the robot to the mean trajectory if ever perturbed. We weigh k_v^p heavily when the the robot position $\mathbf{x}$ is far from any mean trajectory position $\mathbf{x}_i^m$ and vice versa:

$$k_v = \underset{i:||\mathbf{x}-\mathbf{x}_i^m||}{\text{argmin}} \phi(\mathbf{x}, \mathbf{x}_i^m) k_v^a + (1 - \phi(\mathbf{x}, \mathbf{x}_i^m)) k_v^p \tag{10}$$

Where and $\phi(\cdot,\cdot)$ is the radial basis function.

"Perturbed" gain k_v^p is a hyperparameter, and we determine k_v^a via the weighted sum of speeds from states in $\mathbf{x}^m$ based on alignment, as follows:

$$k_v^a = \frac{\sum_{i=1}^{F} P(\tau_k = i|\mathbf{x}^r, \mathbf{x}^m)||\dot{\mathbf{x}}_i^m||}{\sum_{i=1}^{F} P(\tau_k = i|\mathbf{x}^r, \mathbf{x}^m)} \tag{11}$$

IV. Alignment Technique for CALM

The CALM system described above requires an alignment probability distribution $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$ updated for each newly observed state in the robot trajectory. This section describes an HMM alignment strategy capable of providing the desired perturbation and stability behavior for a CALM system. Section IV-A introduces the HMM alignment framework. Section IV-B describes choices of HMM transition functions and how they impact system behavior. Finally, Section IV-C extends CALM's formulation to dynamically follow multiple trajectories while maintaining stability guarantees.

A. Alignment Via HMM

We model alignment distributions $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$ from (5) as a hidden Markov model. We refer to each state in the robot trajectory as an "observation," and alignment is our "hidden" state. Our emission probability is the Gaussian $q_i(\mathbf{x})$ from (3), and our transition probability is $\theta_{j\to i}$ (as we will discuss in Section IV-B). We calculate the robot's alignment probability via the forward algorithm, where $\mathbf{x}_{1:k}^r$ is the robot's trajectory from state 1 to state k:

$$P(\tau_k = i, \mathbf{x}_{1:k}^r|\mathbf{x}^m) = q_i(\mathbf{x}_k^r)$$
$$\times \sum_{j=1}^{F} \theta_{j\to i} P\left(\tau_{k-1} = j, \mathbf{x}_{1:k-1}^r|\mathbf{x}^m\right) \tag{12}$$

Where $P(\tau_{k-1} = j, \mathbf{x}_{1:k-1}^r|\mathbf{x}^m)$ is remembered from the prior observation for every value $j \in \{1, \ldots, F\}$. To retrieve $P(\tau_k = i|\mathbf{x}^r, \mathbf{x}^m)$ for (5), we calculate the following:

$$P(\tau_k = i|\mathbf{x}^r, \mathbf{x}^m) = \frac{P(\tau_k = i, \mathbf{x}^r|\mathbf{x}^m)}{\sum_{i=1}^{F} P(\tau_k = i, \mathbf{x}^r|\mathbf{x}^m)} \tag{13}$$

When calculating the HMM update in (12), we do not use (8) for the transition function $\theta_{j\to i}$. Using the HMM approach, different choices of transition probability grant differing properties, such as convergence to the mean trajectory's final point and ability to be perturbed backwards along the mean trajectory.

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

B. Choice of HMM Transition Function

Transition function $\theta_{j\to i}$ in (12) defines how each new state in $\mathbf{x}^r$ alters its alignment to the mean $\mathbf{x}^m$. For the alignment method presented above to be provably asymptotically stable, we must follow the alignment condition in Theorem 2. To satisfy this, we propose the following transition function:

$$\theta_{j\to i} \propto \begin{cases} \phi(i, j+\Delta) & i > j \\ 1 & i = j = F \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

This is similar to (8), with two differences. First, while (8) requires $\theta_{j\to i} = 0$ when $i < j$ (i.e., each new robot state cannot align to a prior point in the mean trajectory), (14) requires $\theta_{j\to i} = 0$ when $i \leq j$ (i.e., that each new robot state must progress along the mean trajectory by at least one state). Second, (14) states that $\theta_{F\to F} = 1$, meaning once the robot aligns with the mean trajectory's final state, its alignment always remains at that final state. This transition function results in a CALM system stable about $\mathbf{x}^r_F$:

Theorem 3: If a CALM system updates alignment probability $p(\tau_k|\mathbf{x}^r, \mathbf{x}^m)$ via the transition function in (14), it converges to the final point in the mean trajectory: $\mathbf{x} - \mathbf{x}^m_F \to 0$. *(See proof in Appendix C)*

Alternate Transition Function: The transition function in (14) can exhibit realignment when perturbations move the robot forward along the mean $\mathbf{x}^m$. However, it cannot realign to states already passed (i.e. it disallows $\theta_{j\to i} > 0$ when $i \leq j$). A transition function capable of such "backwards" alignment is as follows, for a small value of ϵ:

$$\theta_{j\to i} \propto \begin{cases} \phi(i, j+\Delta) & j \geq i \\ \epsilon & \text{otherwise} \end{cases} \tag{15}$$

Unlike (14), this transition function allows $\theta_{j\to i} > 0$ when $i < j$. While global stability is not guaranteed, in Section V-C we empirically demonstrate consistent convergence while allowing "backwards" perturbations.

C. Accounting for Multiple Demonstrated Clusters

The controller in (1) recreates motions for one demonstrated task. However, TRACER can provide multiple mean trajectories if provided multiple demonstration clusters [19]. We use the HMM above to dynamically chose which mean trajectories to follow while maintaining stability guarantees.

To achieve such behavior, we now assume R TRACER mean trajectories $\{\mathbf{x}^{m_1}, \ldots, \mathbf{x}^{m_R}\}$. Additionally, we define a distribution describing how well the robot trajectory aligns to *any* point in each mean trajectory via marginalizing the HMM joint probability in (12):

$$P(\mathbf{x}^r|\mathbf{x}^{m_i}) = \sum_{j=1}^{F_i} P(\tau_k = j, \mathbf{x}^r|\mathbf{x}^{m_i}) \tag{16}$$

To account for multiple trajectories, we modify (1) to follow the mean trajectory with the highest value of $p(\mathbf{x}^r|\mathbf{x}^{m_i})$:

$$\dot{\mathbf{x}} = \underset{i^*: P(\mathbf{x}^r|\mathbf{x}^{m_i})}{\text{argmax}} \; k_v \frac{\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^{m_{i^*}})}{||\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^{m_{i^*}})||} \tag{17}$$

Theorem 4: If a CALM system uses the transition function in (14), as $k \to \infty$ it will converge to the endpoint of a cluster's mean trajectory: $\exists i \in \{1 \ldots R\}\ \mathbf{x}^r_k - \mathbf{x}^{m_i}_{F_i} \to 0$ *(See proof in Appendix D)*

V. Experiments

In this section, we evaluate CALM on a suite of 2D environments in comparison to baselines with and without perturbations. Additionally, we evaluate CALM in three physical domains on a Franka robot in comparison to select baselines.

We compare against five baselines: Probabilistic Movement Primitives (**ProMP**) [5], Dynamic Movement Primitives (**DMP**) [4], **CLF-NODE** [16], **HSMM** [17], and **LPV-DS** [3]. ProMP, DMP, and CLF-NODE generate time-dependent trajectories (ProMP generates a distribution over trajectories, and DMP and CLF-NODE learn time-dependent controllers) while LPV-DS learns a time-independent dynamical system. HSMM learns a transition function for attractors that is informed by (but not strictly dependent on) time.

A. Replication on 2D Datasets

We evaluate CALM via three illustrative 2D datasets. First, we use *Messy Snake* from Figueroa et al. [3]. Second, we create *Overlap:* four trajectories following a motion that overlaps itself. Finally, we create *Multi-Motion:* six trajectories which follow one of two distinct but overlapping motions (Fig. 3 depicts *Overlap* and *Multi-Motion*).

Each method's performance is illustrated in Fig. 2. For methods that learn a controller (CALM, DMP, LPV-DS, HSMM, and CLF-NODE) we begin at demonstrations' initial point and iteratively roll out a trajectory. For ProMP, we condition the trajectory's first point on demonstrations' initial point, and use the resulting distribution's mean as the predicted trajectory. We evaluate each method qualitatively and via the dynamic time warping distance (DTWD) between each demonstration and a generated motion initialized to the demonstration's initial state. DTWD is a distance measure of trajectories' spatial profiles used by prior work [3]. In MultiMotion, DTWD does not only evaluate methods' ability to follow demonstrations generally, but whether they consistently follow the correct cluster given initial states and perturbations.

We first qualitatively evaluate each method (see Fig. 2):

CALM: CALM's alignment allows it to follow the overlaps in *overlap*, and representing clusters allows it follow each motion in *multi-motion*. TRACER's ability to model intricate demonstrations allows it to follow the twists of *messy-snake*.

LPV-DS: This method's time-independence means that it cannot model the overlaps in *overlap*. Since it does not model trajectory clusters, it starts following the wrong cluster when the two motions in *multi-motion* overlap.

ProMP: ProMP's basis functions struggle to model *messy-snake* (more basis functions help, but performance plateaued at 100). Its time-dependence allows it to model *overlap*, and conditioning on initial state allows it to follow each motion in *multi-motion* (despite no explicit cluster representation).

CLF-NODE: Despite a time-dependent correction, CLF-NODE's underlying time-independent DS cannot model the overlaps in *overlap*. Its underlying neural DS also struggles to capture the twists of *messy-snake*, and its lack of modeling

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

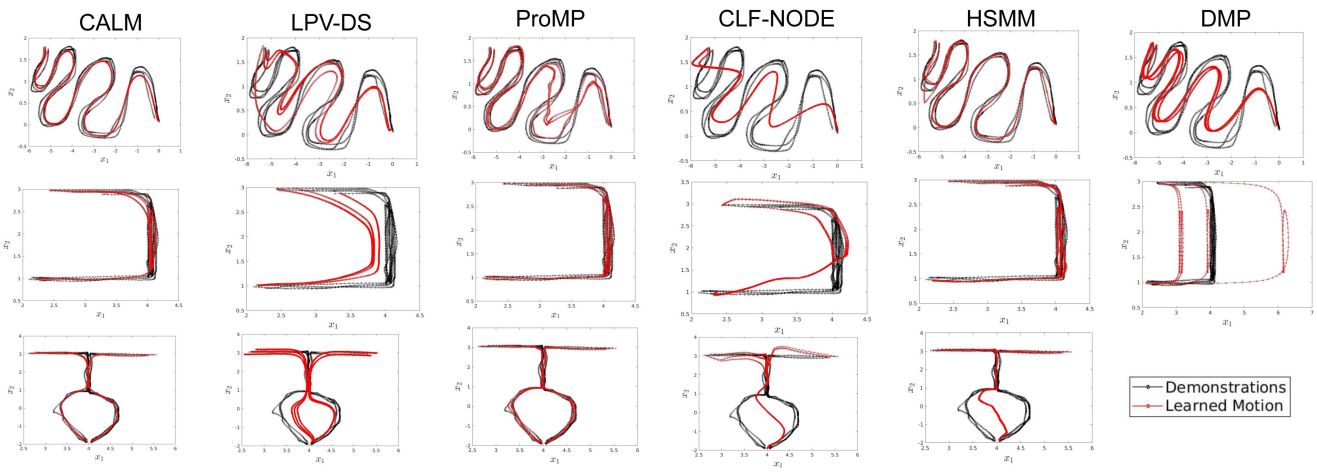

Fig. 2. Behavior of each tested method on three datasets: *Messy Snake* (top), *Overlap* (middle), and *Multi-Motion* (bottom).

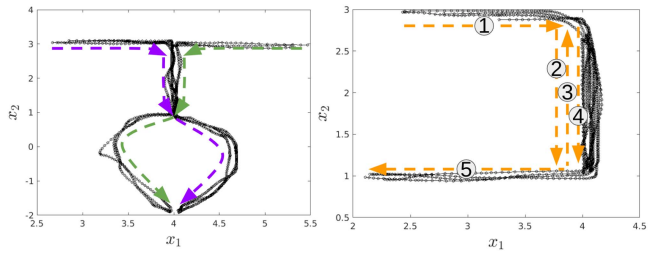

Fig. 3. Two 2D motion datasets created for this letter: *Multi-Motion* (left) and *Overlap* (right). The Multi-Motion dataset includes six demonstrations following two clusters (shown in purple and green); Overlap includes four demonstrations which include a section of overlap (see numbered labels).

TABLE I
MEAN DTWD ACROSS METHODS (DATASET LABELS: *MS* = *MESSY SNAKE*, *O* = *OVERLAP*, *MM* = *MULTI-MOTION*). CALM, PROMP, AND HSMM GENERALLY OUTPERFORM OTHER METHODS.

	CALM	LPV-DS	ProMP	NODE	HSMM	DMP
MS	48.48	196.48	76.34	186.59	46.95	93.19
O	17.12	139.77	8.51	92.11	55.98	310.5
MM	12.77	87.64	8.42	70.24	15.60	NA

trajectory clusters results in a compromise between the two motions in *multi-motion*, cutting between the two.

HSMM: HSMM's time-informed (but not entirely time-dependent) transition function between attractors means that it can model *overlap*, and it can model the twists in *messy-snake* (except a bit around the first turn). However, similar to CLF-NODE, it cuts between the two clusters of *multi-motion* because it does not model demonstration clusters.

DMP: Similar to ProMP, DMPs' basis functions struggle to model the complexity of *messy-snake*. DMPs' forcing function is also sensitive to similar start and end states along one dimension, causing warping along x_1 in *overlap*. DMPs can only learn from one trajectory, and cannot learn *multi-motion* at all (we use TRACER's mean for other datasets).

Table I shows the mean DTWD for each method on each dataset. ProMP, HSMM, and CALM outperform the other methods across all datasets, but CALM outperforms HSMM on *Multi-Motion* and ProMP on *Messy Snake*. On *Overlap* and *Multi-Motion*, CALM's tendency to approach a mean trajectory increased errors compared with ProMP, which is generally able to follow each demonstration's specific trajectory. However, Fig. 2 shows little difference between the generated trajectories in with *Overlap* and *Multi-Motion*.

B. Behavior Under Perturbation

This section compares CALM's behavior to the baselines' when given perturbations. Note that ProMP learns a distribution over trajectories rather than a controller that is iteratively rolled out; thus, ProMP has no defined way to handle a perturbation. For these experiments, we assumed a controller following a trajectory generated from a ProMP would always move toward $\mathbf{x}_i$ at timestep i while matching the generated trajectory's speed at timestep i. Fig. 4 highlights the perturbation behavior for CALM and each baseline. We distinguish "uninformative" perturbations (those not moving the state into a separate region of demonstrations) from "informative" perturbations (those that move the state into a separate region of demonstrations). As the distinction lies in user intent, it is subjective by design – "uninformative" perturbations may involve accidental bumps, while "informative" perturbations represent users intentionally skipping or repeating sections of a trajectory, or changing which trajectory cluster to follow.

All methods except LPV-DS recover from "uninformative" perturbations, as shown in the top row of Fig. 4. This is because the DS tends to model the contours of a trajectory less well when further from demonstrations. The time-dependence inherent to DMP, ProMP, HSMM, and CLF-NODE make them move backward along the demonstrations when given an "informative" perturbation, while the time-independence of CALM and LPV-DS are able to pick up where the perturbation left off. Similarly, for "informative" perturbations on *Multi-Motion* that shift from one demonstration cluster to another, CALM can realign with the new demonstration cluster because it models each cluster explicitly, while ProMP and CLF-NODE do not model each cluster and cannot realign as CALM does.

Fig. 5 depicts methods given unfamiliar initial states. CALM and LPV-DS are time-independent and can begin their motion in the middle of demonstrations near the initial state, while time-dependent methods cannot. DMP's forcing function performs the full motion warped to the initial state. CLF-NODE

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

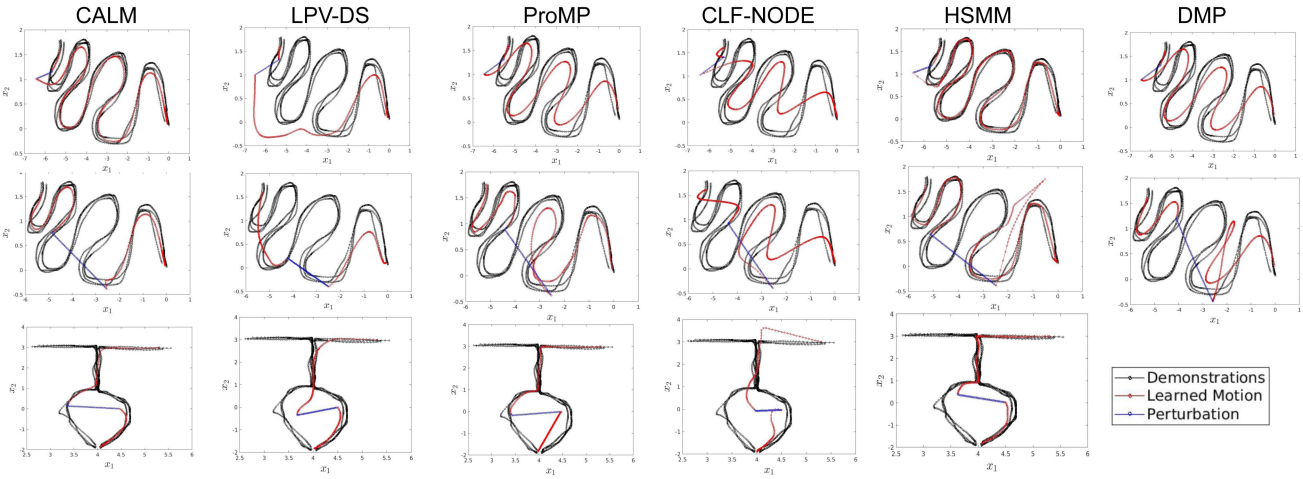

Fig. 4. Behavior under perturbation: *Messy Snake* with "uninformative" perturbation (top), *Messy Snake* with "informative" perturbation to another region of demonstrations (middle), *Multi-Motion* with "informative" perturbation (bottom).

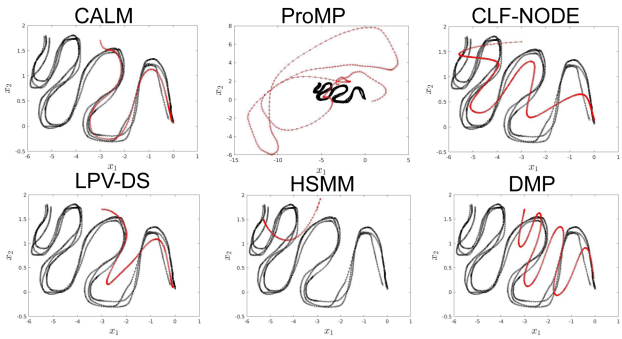

Fig. 5. Each method with an out-of-distribution initial state.

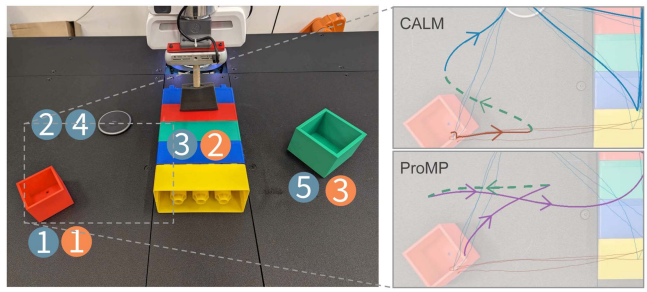

Fig. 6. Left shows our setup and each variant's numbered steps. The "standard" variant (red) starts at the red box, wipes the object, and ends at the green box; "inspection" (blue) stops at the inspection station before and after wipes. Top right shows CALM on a portion of the motion: the robot begins following "standard" (red), is perturbed (green), then realigns to "inspection" (blue). Bottom right shows ProMP: it is unable to distinguish between the variants before or after perturbation, and instead cuts between the two (purple).

and HSMM return to the demonstrations' beginning based on the current timestep, and ProMP does not achieve any coherent imitation of demonstrations. ProMPs can handle conditioning on initial positions near those demonstrated [5]. However, out-of-distribution initial states force unfamiliar trajectories which ProMP conditioning cannot model, causing erratic behavior.

C. Robot Experiments

We apply CALM to three kinesthetically taught tasks on a 7-DoF Franka robot: wiping, writing, and repetitive brushing. The robot uses a compliant PI torque controller using the pseudo-inverse Jacobian to translate desired velocity to torque. We also implement one time-dependent method (ProMP, which had the lowest DTWD in Table I) and one time-independent method (LPV-DS). This letter's associated video depicts the robot's behavior in all tasks.[2]

1) Wiping Task: We provide six demonstrations of a wiping task representative of motions to detect contaminants on an object. We demonstrate two task variants (Fig. 6). In the first, "standard" variant, the robot picks up a brush, wipes the object twice, and moves it to a disposal point. The second variant, "inspection," requires the robot to move the brush to a detector before and after brushing. The variants overlap significantly with themselves and each other, and realignment from intentional perturbation between variants must be maintained for proper task execution. For this task and the writing domain, the robot uses the "backwards" transition function from Section IV-B. Despite no provable convergence to demonstration endpoints, both tasks empirically converge. CALM is able to maintain alignment, and switch between variants when desired. Neither ProMP or LPV-DS represent demonstration clusters, and cannot distinguish the two variants: ProMP combines elements of each, while LPV-DS fails to model "inspection" entirely. LPV-DS's time-independence results in failure to model overlaps in the trajectory, and only wipes the object once (see video).

2) Robot Writing: We teach the robot to write two letters: A and C. Robot writing is a common baseline in LfD [11]. However, we demonstrate in the same region of the state space, showing CALM's capability to maintain alignment in overlapping multi-modal tasks. Fig. 7 shows CALM's reconstruction of each letter and ability to be perturbed from one letter to another. LPV-DS and ProMP's inability to model demonstration clusters renders both unable to write either letter, let alone realign after informative perturbation.

3) Repetitive Brushing: Here, we teach the Franka to dip a brush into a "paint" bowl and brush a box. We teach four

[2]See video at: https://youtu.be/qOFSgZ0K-eo

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

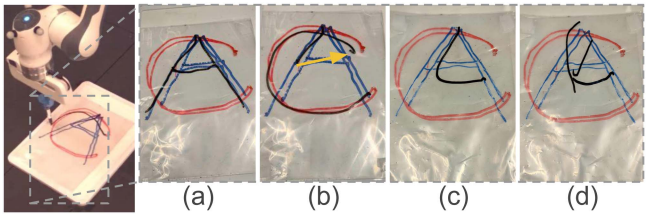

Fig. 7. Demonstrations writing "C" (red), "A" (blue), and the robot motions (black). (a) CALM drawing "A" (b) CALM with perturbation from "A" to "C" (c) LpvDS (d) ProMP.

strokes that are repeated without a defined endpoint (Fig. 8). Learning repeated motions is a common extension of other LfD methods [16], [20], and we use this to demonstrate how modified transition functions can produce desired behavior beyond those described in Section IV-B. For this repeated behavior, we use the modified transition function:

$$\theta_{j\to i} \propto \begin{cases} \phi(i, j+\Delta) & i > j \\ 1 & j = F \text{ and } i = 1 \\ \epsilon & \text{otherwise} \end{cases} \tag{18}$$

CALM consistently maintains alignment to a single stroke, and reliably switches between different strokes with intentional perturbations despite four clusters of demonstrations (as opposed to two in prior tasks). Such behavior demonstrates the flexibility offered by modified transition functions while maintaining desired properties. We do not train LPV-DS or ProMP on this domain, as neither can learn repeated behaviors. See the associated video for example behavior.

VI. Limitations and Future Work

This letter presents a framework for alignment-based LfD; however, several capabilities are left to future work. First, many methods allow for trajectory modulation to satisfy via-points or modified end goals [1], [5]. Future work may take inspiration from Berio et al.'s use of splines in LfD [21]: approximating and manipulating splines on mean trajectories may offer the ability to achieve desired via-point modulation.

Second, we focused on motions in Cartesian space. Prior work has shown the benefits of other coordinate systems in domains such as manipulation [22]. Similar extensions may be a promising direction, but outside this letter's scope.

Lastly, LfD conditioned on sensory data [15] and language input [23] has shown capabilities in more complex behavior. Similar conditioning could inform CALM's choice of mean trajectory and where to align within a mean trajectory.

VII. Conclusion

This letter presented CALM, a learning from demonstrations framework based on alignment. First, we introduced the gradient-based methodology and convergence properties. We then introduced a partial-trajectory alignment technique via HMMs to allow large jumps and convergence to multiple means. Finally, we demonstrated CALM's improvement over baseline methods with regard to tracking, overlapping behavior, and realignment after perturbation on 2D datasets and multiple real robot domains.

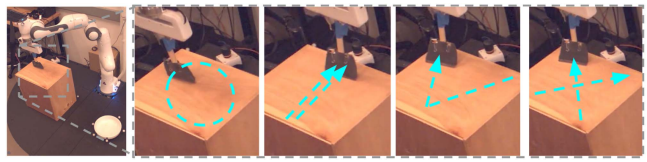

Fig. 8. Four brush strokes learned by CALM. An altered transition function allows for indefinite repetitions of each.

Appendix A
Proofs

A. Theorem 1

Consider (7), rewritten in (19) with $c_i(\mathbf{x}, \tau_{k+1} = i) := q_i(\mathbf{x})P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m)$. Note that $A_i \prec 0 \,\forall i \in \{1, \dots, F\} \Rightarrow \sum A_i = A \prec 0$ (i.e. the sum of negative definite matrices is negative definite) and $\sum c_i > 0$. Hence, the system is always globally asymptotically stable (G.A.S.) around an attractor defined by τ_{k+1}:

$$\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^m) = \sum_{i=1}^{F} \underbrace{\Sigma_m^{-1}}_{\succ 0}(\mathbf{x}_i^m - \mathbf{x}) \underbrace{c_i(\mathbf{x}, \tau_{k+1} = i)}_{\geq 0} \tag{19}$$

$$= \sum_{i=1}^{F} (A_i\mathbf{x} + \mathbf{b}_i) = A\mathbf{x} + \mathbf{b} \tag{20}$$

Note that the global attractor $\mathbf{x}^*$ is:

$$\mathbf{x}^* = -(A^{-1}\mathbf{b}) = \frac{\sum_i c_i \mathbf{x}_i^m}{\sum_i c_i} \tag{21}$$

Thus, if $P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m) = 1$, the attractor $\mathbf{x}^* = \mathbf{x}_i^m$.

B. Theorem 2

If our alignment strategy guarantees $P(\tau_k = F|\mathbf{x}^r, \mathbf{x}^m) = 1$, then as $k \to \infty$, (5) reduces to the following:

$$P(\tau_{k+1} = i|\mathbf{x}^r, \mathbf{x}^m) = \theta_{F\to i} P(\tau_k = F|\mathbf{x}^r, \mathbf{x}^m) \tag{22}$$

If $\theta_{j\to i} = 0 \,\forall i < j$, notice $P(\tau_{k+1} = F|\mathbf{x}^r, \mathbf{x}^m) = 1$. Thus, by (21), the DS in (1) is stable about $\mathbf{x}_F^m$.

C. Theorem 3

We prove this theorem by demonstrating that the condition $P(\tau_k = F|\mathbf{x}^r, \mathbf{x}^m) = 1$ from Theorem 2 is satisfied when using the transition function in (14). We show via induction that a robot trajectory $\mathbf{x}^r$ of length k necessitates that $P(\tau_k = i, \mathbf{x}_{1:k}^r|\mathbf{x}^m) = 0 \,\forall i < k$:

Base case: Assume $P(\tau_{k-1} = i, \mathbf{x}_{1:k-1}^r|\mathbf{x}^m)$ is non-zero for all values $i = \{1, \dots, F\}$. Using the transition function from (14), we have $\theta_{i\to 1} = 0$ for any value i. Therefore, by (12), $P(\tau_k = 1, \mathbf{x}_{1:k}^r|\mathbf{x}^m) = 0$.

Inductive Case: Assume $P(\tau_{k-1} = j, \mathbf{x}_{1:k-1}^r|\mathbf{x}^m) = 0 \,\forall j < v$ for some $v \geq 1$. In the next HMM update, we can remove the first v elements of the sum in (12), leaving:

$$P(\tau_k = i, \mathbf{x}_{1:k}^r|\mathbf{x}^m) = q_i(\mathbf{x}_k^r) \sum_{j=v}^{F} \theta_{j\to i} P\left(\tau_{k-1} = j, \mathbf{x}_{1:k-1}^r|\mathbf{x}^m\right) \tag{23}$$

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

For any $j \in \{v, \ldots, F\}$, $\theta_{j \to i} = 0$ when $i < v+1$. Thus, $P(\tau_k = i, \mathbf{x}^r_{1:k} | \mathbf{x}^m) = 0 \; \forall i < v+1$, completing the induction.

Once $v = F$, we have $P(\tau_{k-1} = j, \mathbf{x}^r | \mathbf{x}^m) = 0 \; \forall j < F$, leaving the HMM update as follows:

$$P(\tau_k = i, \mathbf{x}^r | \mathbf{x}^m) = q_i(\mathbf{x}^r_k) \theta_{F \to i} P(\tau_{k-1} = F, \mathbf{x}^r_{1:k-1} | \mathbf{x}^m) \tag{24}$$

Since $\theta_{F \to F} = 1$, $P(\tau_k = i, \mathbf{x}^r | \mathbf{x}^m) = 0 \; \forall i \neq F$. Via (13), $P(\tau_k = F | \mathbf{x}^r, \mathbf{x}^m) = 1$, satisfying the condition in Theorem 2.

D. Theorem 4

Lemma 1: As $k \to \infty$, $P(\mathbf{x}^r_{1:k+1} | \mathbf{x}^{m_i}) = q_{F_i}(\mathbf{x}^r_{k+1}) P(\mathbf{x}^r_{1:k} | \mathbf{x}^{m_i})$ (24) in Theorem 3 shows that as $k \to \infty$:

$$P(\tau_{k+1} = j, \mathbf{x}^r | \mathbf{x}^{m_i}) = q_j(\mathbf{x}^r_{k+1}) \theta_{F_i \to j} P(\tau_k = F_i, \mathbf{x}^r_{1:k} | \mathbf{x}^{m_i}) \tag{25}$$

$$= \begin{cases} q_j(\mathbf{x}^r_k) P(\tau_k = j, \mathbf{x}^r_{1:k} | \mathbf{x}^m) & j = F_i \\ 0 & \text{Otherwise} \end{cases} \tag{26}$$

Where step 2 uses (14). Substituting into (16) and observing via Theorem 3 that $P(\mathbf{x}^r_{1:k} | \mathbf{x}^{m_i}) = P(\tau_k = F_i, \mathbf{x}^r_{1:k} | \mathbf{x}^{m_i})$:

$$P(\mathbf{x}^r_{1:k+1} | \mathbf{x}^{m_i}) = q_{F_i}(\mathbf{x}^r_k) P(\tau_k = F_i, \mathbf{x}^r_{1:k} | \mathbf{x}^{m_i}) \tag{27}$$

$$= q_{F_i}(\mathbf{x}^r_k) P(\mathbf{x}^r_{1:k} | \mathbf{x}^{m_i}) \tag{28}$$

□

Lemma 2: As $k \to \infty, \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^{m_i}) \propto \nabla_{\mathbf{x}} q_{F_i}(\mathbf{x})$ proof: Using Theorem 2, as $k \to \infty$, (2) becomes:

$$g(\mathbf{x}, \mathbf{x}^r, \mathbf{x}^{m_i}) = q_{F_i}(\mathbf{x}) p(\tau_{k+1} = F_i | \mathbf{x}^r, \mathbf{x}^{m_i}) \tag{29}$$

□

Let i^* be the index of the mean trajectory for which:

$$p(\mathbf{x}^r | \mathbf{x}^{m_{i^*}}) > p(\mathbf{x}^r | \mathbf{x}^{m_i}) \; \forall i \neq i^* \tag{30}$$

At time $k-1$, let $i^* = i'$. At time k, i^* can either remain the same ($i^* = i'$) or change ($i^* \neq i'$). If i^* remains the same, (1) and Lemma 2 indicate $q_{F_{i'}}(\mathbf{x}^r_k) \geq q_{F_{i'}}(\mathbf{x}^r_{k-1})$. Equality only occurs at the max of $q_{F_{i'}}(\cdot)$ where $\nabla q_{F_{i'}}(\mathbf{x}^r_k) = 0$:

$$\mathbf{x}^r_k = \mathbf{x}^{m_{i'}}_{F_{i'}} \tag{31}$$

Now imagine at time k, $i^* = i''$ (i^* changes). Here, $\mathbf{x}^{m_{i''}}$ "overtakes" $\mathbf{x}^{m_{i'}}$ so that $p(\mathbf{x}^r | \mathbf{x}^{m_{i''}}) > p(\mathbf{x}^r | \mathbf{x}^{m_{i'}})$. Notice by Lemma 1, this "overtaking" requires $q_{F_{i''}}(\mathbf{x}^r_k) > q_{F_{i'}}(\mathbf{x}^r_k)$. Therefore, whether i^* changes or not, $q_{F_{i^*}}(\mathbf{x}^r_k)$ increases at every step. Thus, as k increases, $\mathbf{x}^r_k$ must eventually reach a max of $q_{F_{i^*}}(\cdot)$ where $\nabla q_{F_{i^*}}(\mathbf{x}^r_k) = 0$, as in (31). By Lemma 2, $\nabla q_{F_{i^*}}(\mathbf{x}^r_k) = 0 \Rightarrow \nabla g(\mathbf{x}^r_k, \mathbf{x}^r, \mathbf{x}^{m_i}) = 0$. Thus, ∇g is stable at point $\mathbf{x}^{m_i}_{F_i}$ for some i, proving the theorem.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[2] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[3] N. B. Figueroa Fernandez and A. Billard, "A physically-consistent Bayesian non-parametric mixture model for dynamical system learning," in *Proc. Mach. Learn. Res.*, 2018, pp. 927–946.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, vol. 26, pp 2616–2624.

[6] S. Dixon, "Live tracking of musical performances using on-line time warping," in *Proc. 8th Int. Conf. Digit. Audio Effects*, 2005, vol. 92, pp. 92–97.

[7] P. A. Lasota and J. A. Shah, "Bayesian estimator for partial trajectory alignment," in *Proc. Robot., Sci. Syst.*, 2019.

[8] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, vol. 44, pp 1684–1704, 2023.

[9] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Proc. Robot., Sci. Syst.*, Daegu, Republic of Korea, Jul. 2023.

[10] A. Mandlekar et al., "What matters in learning from offline human demonstrations for robot manipulation," in *Proc. 5th Conf. Robot Learn.*, in Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., Nov. 08–11, 2022, vol. 164, pp. 1678–1690.

[11] N. Figueroa and A. Billard, "Locally active globally stable dynamical systems: Theory, learning, and experiments," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 312–347, 2022.

[12] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil, "Neural learning of vector fields for encoding stable dynamical systems," *Neurocomputing*, vol. 141, pp. 3–14, 2014.

[13] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5231–5237.

[14] T. Kulak, J. Silvério, and S. Calinon, "Fourier movement primitives: An approach for learning rhythmic robot skills from demonstrations," in *Proc. Robot., Sci. Syst.*, 2020.

[15] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "Prodmp: A unified perspective on dynamic and probabilistic movement primitives," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2325–2332, Apr. 2023.

[16] F. Nawaz, T. Li, N. Matni, and N. Figueroa, "Learning complex motion plans using neural odes with safety and stability guarantees," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 17216–17222.

[17] S. Calinon, A. Pistillo, and D. G. Caldwell, "Encoding the time and space constraints of a task in explicit-duration hidden Markov model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 3413–3418.

[18] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robot. Automat. Mag.*, vol. 17, no. 2, pp. 44–54, Jun. 2010.

[19] C. Fourie, "Real-time anticipation and entrainment in human-robot interaction," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2024.

[20] T. Petrič, A. Gams, L. Žlajpah, and A. Ude, "Online learning of task-specific dynamics for periodic tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1790–1795.

[21] D. Berio, S. Calinon, and F. F. Leymarie, "Generating calligraphic trajectories with model predictive control," in *Graphics Interface*, ACM, May 2017.

[22] B. Ti, A. Razmjoo, Y. Gao, J. Zhao, and S. Calinon, "A geometric optimal control approach for imitation and generalization of manipulation skills," *Robot. Auton. Syst.*, vol. 164, 2023.

[23] S. Sharan et al., "Plan diffuser: Grounding LLM planners with diffusion models for robotic manipulation," in *Proc. Bridging Gap Cogn. Sci. Robot Learn. Real World, Progresses New Directions*, 2024.

Authorized licensed use limited to: MIT. Downloaded on March 09,2026 at 03:50:47 UTC from IEEE Xplore. Restrictions apply.