2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning Constraint-Aware Dynamical Systems from Human Demonstrations for Constrained Manipulation Tasks

Soyoun Sung and Keehoon Kim, *Senior Member, IEEE,*

***Abstract*— Learning from demonstration (LfD) enables robots to acquire new skills from human examples without explicit programming. Dynamical system (DS)-based approaches, in particular, have shown robustness to disturbances and adaptability in unstructured environments. However, existing methods often fail to incorporate task-specific constraints—such as grasp locations, execution starting points, or motion restrictions—that are critical for reliable execution. This limitation becomes especially problematic in tool-use scenarios, where both the environment and the grasped tool impose strict restrictions on feasible motions. To address this challenge, we propose a novel constraint-aware DS framework that automatically extracts and encodes task-specific constraints directly from demonstration data. The key idea is that task-critical configurations, repeatedly observed across successful demonstrations, can be identified and modeled as essential regions for task success using Gaussian Process Regression. By embedding these constraints, the proposed method generates motions that remain robust to environmental variations and tool-induced limitations. Experiments with a 7-DoF robotic manipulator demonstrate that our framework significantly improves task success rates over state-of-the-art methods. Real-world evaluations on daily-life tasks, such as dishware collection, further confirm its practicality and potential for real-world robotic applications.**

I. Introduction

Learning from demonstration (LfD) has become a widely used paradigm in robotics, as it allows robots to acquire new skills from human demonstrations without requiring explicit programming or detailed control instructions [1]. In contrast to conventional robot programming approaches that demand expert-level knowledge, LfD enables even non-expert users to teach robots effectively.

The primary goal of LfD is not merely to replicate demonstrated actions but to enable robots to operate in unstructured and dynamic environments. Pure trajectory replay is often insufficient, as it lacks adaptability to variations in the environment [2], [3]. In practical scenarios, robots may need to start from novel states, handle modified targets, or cope with unexpected external disturbances, such as human intervention. Simply retracing the demonstrated trajectory can therefore result in inefficiency and failure. Furthermore, demonstration data—particularly those gathered via teleoperation—may contain redundant or noisy segments. Hence, LfD should emphasize extracting high-level, task-relevant structures from demonstrations, focusing on *what* to learn rather than blindly imitating low-level motions.

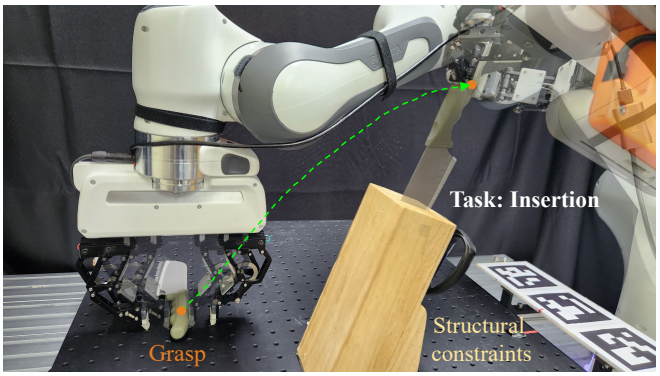

Fig. 1. Example of a tool-held constrained task (knife insertion), where the robot must grasp the tool and perform the insertion under strong structural constraints imposed by the environment.

Two main paradigms exist in LfD. The first represents trajectories as explicit functions of time, as seen in methods such as dynamic movement primitives (DMP) [4], probabilistic movement primitives (ProMP) [5], kernelized movement primitives (KMP) [6], and structured prediction approaches [7]. The second formulates trajectories as state-dependent representations, where robot states—such as end-effector position and orientation—determine the reproduced trajectory [8].

This work focuses on the state-dependent paradigm, which is more robust to spatial and temporal perturbations[1]. Within this category, autonomous dynamical systems (DS) are of particular interest, as they encode demonstrations as time-invariant velocity fields. DS-based methods have shown strong adaptability in diverse contexts, including robustness to disturbances [8], human-robot shared control [9], adaptability to moving targets [10], and obstacle avoidance [11].

Tasks encoded using DS as LfD can be broadly categorized into two groups. Free-motion tasks, such as pick-and-place, are executed without structural restrictions. In contrast, constrained tasks involve motions limited by environmental geometry, such as opening a drawer, closing a door, assembly, peg-in-hole, or screwing. A subset of constrained tasks is considered highly constrained, where task-specific structures restrict three or more degrees of freedom of the robot for successful task performance.

In addition, tool usage plays an important role in shap-

This work was supported by the Robot Industrial Core Technology Development Project (20018745, Development of dismantling work technology for recycling of variety EV battery packs with human-robot cooperation) funded By the Ministry of Trade, Industry Energy(MOTIE, Korea).

All authors are with Department of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), 37673, Gyeong-buk, South Korea. Email: {sungsy1206, khk}@postech.ac.kr

[1]For instance, a human may interrupt the robot mid-task, prolonging or shortening the required execution time compared to the original demonstration.

ing task difficulty. Highly constrained tasks can be further divided into (i) tool-attached constrained tasks, where a tool is permanently mounted on the robot's end-effector, and (ii) tool-held constrained tasks, where the robot must first grasp and orient a tool before task execution. Everyday tasks such as stacking cups, placing plates into a plate rack, inserting chopsticks, or placing a knife into a holder belong to the second category. Tool-held constrained tasks are often more restrictive, as both the environment and the grasped tool impose constraints, and task success critically depends on grasp configuration.

Several state-dependent velocity field-based (DS-based) methods have been proposed to encode demonstrations. The Stable Estimator of Dynamical Systems (SEDS) [8] employs a Gaussian mixture model to guarantee convergence to a globally stable equilibrium while reproducing demonstrated motions. This method provided a principled way to ensure both stability and fidelity to demonstrations, establishing the foundation for DS-based learning from demonstration.

Building on this idea, the Linear Parameter-Varying DS (LPV-DS) framework [12] was later introduced, where velocity-aware clustering was leveraged to construct locally consistent models capable of capturing more complex motion patterns. This framework enhanced the expressiveness of DS-based methods while preserving the stability guarantees of SEDS.

More recently, a state-dependent motion generator was proposed in [9], which directs the robot toward demonstration states most consistent with its current configuration. This approach enables the reproduction of intricate trajectory patterns and provides improved adaptability in dynamic environments, further extending the applicability of DS-based approaches.

Despite these advances, existing methods face important limitations. First, reproducing motions based solely on Euclidean closeness to demonstrated states does not always guarantee task success, particularly when the task-specific motion constraints impose a tight coupling between position and orientation. Second, most approaches neglect to explicitly model task-specific motion constraints—such as tool grasping location, task execution starting point, and the desired constrained motion—even though demonstrations inherently reflect them [8]–[10], [12]–[16]. This omission often leads to reduced reliability in highly constrained tasks.

To address these issues, this paper introduces a constraint-aware dynamical system framework that explicitly extracts and encodes task constraints from demonstrations. The proposed approach is designed to improve success rates in highly constrained tasks, with particular emphasis on tool-held tasks, where grasp configuration and tool usage critically influence performance.

II. Method

The main idea of the proposed framework is that task-specific motion constraints critical for task success tend to appear repeatedly within the demonstrated trajectories of a given task. To capture these constraints directly from demonstration data, we begin by introducing several key terms. Configuration data includes both position and orientation, and Fig. 2 illustrates the defined concepts.

1) **Graspable configuration (GC)**: Relative robot–tool configurations observed during stable grasps in successful task executions.
2) **Critical configuration route (CC)**: Segments of configuration trajectories that play a decisive role in achieving task success within demonstrated executions.
3) **Non-critical configuration route (NCC)**: Portions of the demonstrated trajectories that are not selected as CC and thus do not directly contribute to task success.
4) **Critical configuration region (CCR)**: A configuration region derived from CC data that characterizes essential states for task accomplishment. This region encompasses multiple feasible trajectories leading to success, even if they are not explicitly demonstrated.
5) **Critical configuration trajectory (CCT)**: A trajectory estimated from the CCR that represents the desired configuration evolution. The CCT provides a reference path that can guide the robot toward successful execution.

TABLE I
Comparison with State-of-the-Art Methods

Contents	Proposed	SEDS [8]	LPV-DS [12]	Mean trajectory [9]
Time invariant	✓	✓	✓	✓
Nonlinear trajectory	✓	✓	✓	✓
Highly constrained task *Position*	✓	×	× or (✓)	(✓)
Highly constrained task *Position-Orientation*	✓	×	×	×
Tool-held constrained task	✓	–	–	–

Bold entries indicate the main contributions of this work. (–) denotes that the corresponding aspect is not addressed in the paper.

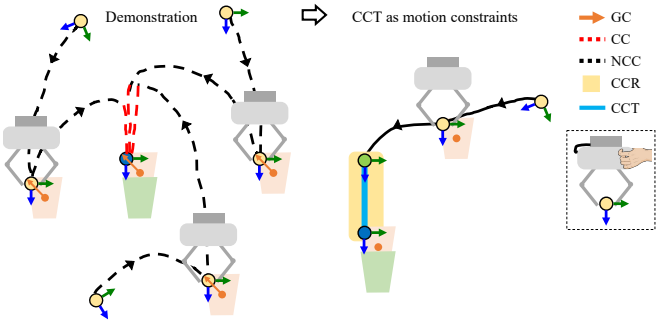

Fig. 2. This figure illustrates the defined terms—GC, CC, NCC, CCR, and CCT—for the cup-stacking task. The dotted line shows demonstration data, and the solid line represents the reproduced motion. The blue dot denotes the demonstrated target state, and the green dot is the estimated via-point from eq (5). During the demonstration, the robot reaches the cup, grasps it, and converges to the via-point to perform the stacking task under motion constraints learned by the GPR model.

The proposed generates the constraint-aware DS to encode human demonstrations, which enables the execution of highly constrained tasks. Methods such as selecting CC, and modeling motion constraints, including grasp-point, via-point

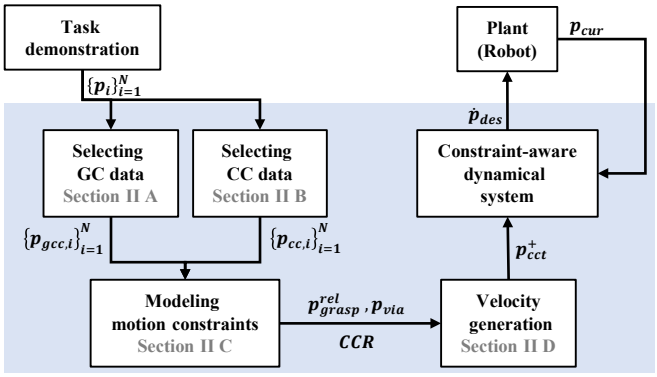

Fig. 3. The overall flow of the proposed method is shown. The framework involves generating a constraint-aware autonomous dynamical system (DS) to execute the demonstrated task subject to stringent constraints. Task-specific constraints contain the grasp-point, via-point, and motion constraints. For the description of each input and output of the block, please see Table II and section II.

and task-specific constraints, are introduced. Finally, the desired velocity using the modeled constraints is explained to conduct the highly constrained tasks successfully. The overall flow of the proposed method is shown in Fig. 3. Also, Table II explains the notations used in this paper.

TABLE II
NOMENCLATURES

Symbol	Description
$\{s\}$	World frame fixed to the robot ground origin
$\{t\}$	Target frame fixed to the demonstrated task target
$[.]_{\mathrm{cur}}$	Current robot state
$[.]_{\mathrm{tar}}$	Task target state
$[.]_{\mathrm{cc}}$	Selected critical configuration route (CC) data
$[.]_{\mathrm{ncc}}$	Non- critical configuration route (NCC) data
$[.]^{\mathrm{rel}}$	State relative to the target state in $\{s\}$ (position) and in $\{t\}$ (orientation)
$[.]^{\mathrm{rel}}_{\mathrm{grasp}}$	Estimated grasp-point state from eq (5) relative to the object tool
$[.]^{\mathrm{rel}}_{\mathrm{via}}$	Calculated via-point state from eq (5)
$[.]_{\mathrm{via}}$	Via-point state in $\{s\}$
$[.]_{\mathrm{gpr,via}}$	Via-point estimated from GPR model Input to GPR: error between $\boldsymbol{p}_{\mathrm{tar}}$ and $\boldsymbol{p}_{\mathrm{via}}$
$[.]^{\mathrm{rel}}_{\mathrm{cct}}$	Relative CCT estimated from GPR model Input to GPR: error between $\boldsymbol{p}_{\mathrm{tar}}$ and $\boldsymbol{p}_{\mathrm{cur}}$
$[.]^{\mathrm{rev}}_{\mathrm{cct}}$	Relative reversed CCT estimated from reverse GPR model
$[.]_{\mathrm{cct}}$	CCT estimated from GPR model Transformed to spatial frame from $[.]^{\mathrm{rel}}_{\mathrm{cct}}$
$[.]^+$	Next state when the robot moves linearly by a one-time step from the $[.]_{\mathrm{cur}}$ to the $[.]_{\mathrm{tar}}$
$[.]^{\mathrm{rel},+}_{\mathrm{cct}}$	Relative CCT estimated from GPR model Input to GPR: error between $\boldsymbol{p}_{\mathrm{tar}}$ and $\boldsymbol{p}^+$
$[.]^+_{\mathrm{cct}}$	CCT estimated from GPR model Transformed to spatial frame from $[.]^{\mathrm{rel},+}_{\mathrm{cct}}$
$[.]_{\mathrm{des}}$	Desired state of next time step in $\{s\}$

[.] denotes $\boldsymbol{p}$, $\boldsymbol{r}$, $\boldsymbol{\xi}$, $\boldsymbol{R}\in SO(3)$, or $\boldsymbol{T}=(\boldsymbol{R},\boldsymbol{r})\in SE(3)$, where $SE(3)$ is the Special Euclidean group of homogeneous transformation matrices.

A. Selecting graspable configuration (GC)

It is crucial to determine, from demonstration data, which part of the tool the robot grasped while performing a task. If grasping points are selected arbitrarily, it becomes difficult to define motion constraints that guarantee successful task execution based on the demonstrated trajectories. This issue is particularly significant for highly constrained tasks.

Therefore, in this section, we propose a method to select configuration data corresponding to stable grasps of the tool. The underlying idea is that, once the robot grasps the tool stably, the relative configuration between the tool and the robot remains almost constant. Based on this observation, we define the method for selecting graspable configurations (GC) as follows.

For the i-th trajectory of the robot $\{\boldsymbol{p}_{i_j}\}_{j=1}^t$ with $\boldsymbol{p}_{i_j} = [\boldsymbol{r}_{i_j} \quad \boldsymbol{\xi}_{i_j}]^\top$ and $\boldsymbol{\xi}_{i_j} = \log(\boldsymbol{R}_{i_j})$[2], and the i-th trajectory of the tool $\{\boldsymbol{p}^{\mathrm{tool}}_{i_j}\}_{j=1}^t$, the relative configuration of the robot based on the tool frame is defined as

$$\boldsymbol{r}^{\mathrm{rel,tool}}_{i_j} = \boldsymbol{R}^{\mathrm{tool}\top}_{i_j}\left(\boldsymbol{r}^{\mathrm{tool}}_{i_j} - \boldsymbol{r}_{i_j}\right), \quad \boldsymbol{R}^{\mathrm{rel,tool}}_{i_j} = \boldsymbol{R}^{\mathrm{tool}\top}_{i_j}\boldsymbol{R}_{i_j}.$$

Then, the configuration data are selected as GC as follows with a small constant ϵ. Given a window size $N \in \mathbb{N}$,[3] define the index set $\mathcal{K}_j \coloneqq \{1, 2, \ldots, \min(N,\, t-j)\}$.

$$\begin{cases} \Delta\boldsymbol{p}^{\mathrm{rel,\,tool}}_{i,\,j\to k} \;=\; \boldsymbol{p}^{\mathrm{rel,\,tool}}_{i\,j+k} - \boldsymbol{p}^{\mathrm{rel,\,tool}}_{i\,j}, \quad k \in \mathcal{K}_j, & \text{(1a)} \\ D_{i,j} \;\coloneqq\; \max\limits_{k\in\mathcal{K}_j}\left\|\Delta\boldsymbol{p}^{\mathrm{rel,\,tool}}_{i,\,j\to k}\right\| - \min\limits_{k\in\mathcal{K}_j}\left\|\Delta\boldsymbol{p}^{\mathrm{rel,\,tool}}_{i,\,j\to k}\right\|, & \text{(1b)} \\ \boldsymbol{p}^{\mathrm{rel,\,tool}}_{i_j} \in \{\boldsymbol{GC}\}_i \quad \text{if} \quad D_{i,j} \le \epsilon. & \text{(1c)} \end{cases}$$

B. Selecting critical configuration route (CC)

The demonstration data are transformed into relative states with respect to the target configuration of each trajectory, so that all demonstrations converge to the common target $(\boldsymbol{0}_{3\times1}, \boldsymbol{I}_3)$. For the i-th trajectory $\{\boldsymbol{p}_{i_j}\}_{j=1}^t$, the relative position and orientation are defined as

$$\boldsymbol{r}^{\mathrm{rel}}_{i_j} = \boldsymbol{r}_{i_t} - \boldsymbol{r}_{i_j}, \quad \boldsymbol{R}^{\mathrm{rel}}_{i_j} = \boldsymbol{R}^\top_{i_j}\boldsymbol{R}_{i_t}.$$

This preprocessing reduces variance among demonstrations and enables generalization to varying targets, while the transformation $T(\boldsymbol{p}^{\mathrm{rel}}) = \boldsymbol{p}$ maps the relative states back to the world frame.

$$T(\boldsymbol{r}^{\mathrm{rel}}, \boldsymbol{\xi}^{\mathrm{rel}}) = \begin{cases} \boldsymbol{r} = \boldsymbol{r}_{\mathrm{tar}} - \boldsymbol{r}^{\mathrm{rel}} \\ \boldsymbol{\xi} = \log(\boldsymbol{R}_{\mathrm{tar}}\left(\exp(\boldsymbol{\xi}^{\mathrm{rel}})\right)^\top) \end{cases} \tag{2}$$

Before modeling task-specific motion constraints (CCR, CCT), it is necessary to extract critical configuration routes (CCs), defined as frequently repeated trajectories with small variance in demonstrations, based on the idea that task-relevant routes tend to be consistently reproduced. The

[2]For $\boldsymbol{R} \in SO(3)$ and $\boldsymbol{\xi} \in \mathfrak{so}(3) \cong \mathbb{R}^3$, the logarithm and exponential maps establish a local diffeomorphism between $SO(3)$ and its Lie algebra: $\boldsymbol{\xi} = \log(\boldsymbol{R})$ and $\boldsymbol{R} = \exp(\boldsymbol{\xi})$, which are inverse mappings of each other.

[3]Once the tool is grasped, it is assumed to remain grasped until task completion. The window size N is selected according to the data sampling frequency to detect a sustained grasp.

selection of CCs proceeds in two steps: (i) threshold bound calculation and (ii) CC data selection.

Threshold bound calculation: A reference threshold d_{bound} is determined using both successful and fail demonstrations[4]. The underlying concept is to quantify a representative measure of the difference between successful and fail demonstrations. Let $\boldsymbol{p}_{i_k}$ and $\boldsymbol{p}_{j_n}$ denote the k-th and n-th points of the i-th success and j-th fail demonstrations, respectively. Then the threshold bound is given by

$$d_{\text{bound}} = \max_i \max_j \min_k \min_n \|\boldsymbol{p}_{i_k} - \boldsymbol{p}_{j_n}\|_2. \tag{3}$$

This value serves as the upper bound for selecting CCs in the next step.

CC data selection: At this stage, the reference value derived previously is applied to the full set of successful demonstration data to determine the CCs. To quantify the difference between sets of demonstrations across trials, the following method is proposed with the threshold value d_{thrd}.

$$\begin{cases} d_{i_k,\max} = \max_j \min_n \|\boldsymbol{p}_{i_k} - \boldsymbol{p}_{j_n}\|_2 & \text{(4a)} \\ \boldsymbol{p}_{i_k} \in \{\boldsymbol{CC}\}_i \quad \text{if } d_{i_k,\max} \le d_{\text{thrd}} \le d_{\text{bound}} & \text{(4b)} \end{cases}$$

Although the same algorithm can be used with data represented as $[\boldsymbol{r} \;\; \boldsymbol{q}]^\top$ with quaternion $\boldsymbol{q}$, this work adopts the Lie algebra $\boldsymbol{\xi}$ representation, which not only reduces dimensionality but also ensures the reversibility of the modeled motion constraints (see Section II-C).

C. Modeling motion constraints for successful task execution

Graspable configurations (GCs) and critical configuration routes (CCs) extracted from demonstration data provide the basis for modeling task-relevant constraints. The modeling consists of three components: the grasp-point, defining where the robot grips the tool; the via-point, indicating where motion constraint adherence begins; and the task-specific motion constraints.

Grasp-point and Via-point Estimation: The grasp-point corresponds to a stable tool-grasping configuration required before task execution, while the via-point denotes the configuration at which motion constraint adherence starts after grasping. Both are computed as weighted centroids of the initial samples of the respective datasets: the grasp-point from all GCs $\{\{\boldsymbol{GC}\}_i\}_{i=1}^N$ and the via-point from all CCs $\{\{\boldsymbol{CC}\}_i\}_{i=1}^N$. Let $\boldsymbol{\lambda}_i$ denote the first element of $\{\boldsymbol{GC}\}_i$ or $\{\boldsymbol{CC}\}_i$. Then,

$$\boldsymbol{\lambda}^{\text{rel}} = \frac{\sum_{n=1}^N W_\lambda \boldsymbol{\lambda}_n}{\sum_{n=1}^N W_\lambda}, \quad W_\lambda = \frac{1}{\sqrt{\sum_{i=1, i\neq n}^N \|\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_i\|^2}}. \tag{5}$$

Here, $\boldsymbol{\lambda} \in \{\boldsymbol{r}, \boldsymbol{\xi}\}$ denotes the relative robot state, defined with respect to the detected tool for the grasp-point and to the target state for the via-point. The global via-point is obtained as $\boldsymbol{p}_{\text{via}} = T(\boldsymbol{p}_{\text{via}}^{\text{rel}})$. Unlike conventional approaches that manually specify these points [7], [9], both are directly inferred from demonstration data.

Task-specific motion constraints modeling: In this step, motion constraints that ensure successful task completion are derived using Gaussian Process Regression (GPR). The two main components of these constraints are the critical configuration region (CCR), representing the spatial region relevant for the task, and the critical configuration trajectory (CCT), specifying the desired relative state at the next time step. To model this, separate GPR models are trained for each degree of freedom in $\boldsymbol{p} = [r_x, r_y, r_z, \xi_x, \xi_y, \xi_z]^\top$. The inputs to each model consist of the position and orientation deviations from the target, while the outputs provide the predicted relative next state, which defines the CCT.

$$\boldsymbol{p}_{\text{cct}}^{\text{rel}} = GPR(\|\boldsymbol{r}_{\text{cur}} - \boldsymbol{r}_{\text{tar}}\|, \|\log(\boldsymbol{R}_{\text{cur}}^\top \boldsymbol{R}_{\text{tar}})\|). \tag{6}$$

CCR can be determined using the mean and variance of the GPR predictions, with the flexibility to adjust the restriction through Z-scores.

Reversibility of constraints: The modeled motion constraints can be applied in the reverse direction. For example, constraints learned from a door-closing demonstration can be inverted to perform the door-opening task. This reversibility is achieved by negating the output of the original GPR in eq (6):

$$\boldsymbol{p}_{\text{cct}}^{\text{rev}} = -\boldsymbol{p}_{\text{cct}}^{\text{rel}}. \tag{7}$$

The reversibility relies on representing orientations in the Lie algebra. An exponential coordinate $\boldsymbol{\xi}$ can be expressed as $\boldsymbol{\xi} = \hat{\boldsymbol{\xi}}\|\boldsymbol{\xi}\|$, where $\hat{\boldsymbol{\xi}}$ is the screw axis and $\|\boldsymbol{\xi}\|$ is the rotation angle. Negating $\boldsymbol{\xi}$ preserves the rotation axis while reversing the rotation direction, thereby enabling reverse motion. In contrast, quaternions cannot be negated to reverse a rotation.

Proof: Let $\boldsymbol{v}_1 = \boldsymbol{q}_1 \boldsymbol{v} \boldsymbol{q}_1^{-1}$ and $\boldsymbol{v}_2 = \boldsymbol{q}_2 \boldsymbol{v} \boldsymbol{q}_2^{-1}$ be rotations of $\boldsymbol{v}$ by quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, respectively. If $\boldsymbol{q}_2 = -\boldsymbol{q}_1$, then $\boldsymbol{v}_1 = \boldsymbol{v}_2$, implying that negating a quaternion does not invert the rotation.

Therefore, Lie algebra is employed not only for compact representation but also to enable reversible motion constraints, allowing the robot to follow learned constraints in both forward and reverse directions.

Consequently, the constraints embedded in the demonstration data can be modeled to enable successful task execution. This is accomplished by identifying the grasp-point $\boldsymbol{p}_{\text{grasp}}^{\text{rel}}$ at which the robot holds the tool, the via-point $\boldsymbol{p}_{\text{via}}$ that the robot must pass, and the expected relative state $\boldsymbol{p}_{\text{cct}}^{\text{rel}}$ that defines the desired configuration for task completion.

D. Generating desired velocity with activation condition

Rather than continuously enforcing the robot to follow the motion constraints represented by $\boldsymbol{p}_{\text{cct}}$, the desired state and velocity predicted by the GPR model should be applied selectively, only when specific conditions for successful task execution are met. The activation criteria are derived from parameters and vectors extracted from the demonstration data.

[4]A fail demonstration refers to a case where the demonstrator intentionally prevents the robot from following task-specific constraints near the constraint region.

Algorithm 1 Constraints-aware dynamical system for highly constrained tasks

Require:

$r_{\text{err,cct}} = \|\boldsymbol{r}_{\text{cct}} - \boldsymbol{r}_{\text{cur}}\|$, $\xi_{\text{err,cct}} = \|\log(\boldsymbol{R}_{\text{cur}}^{\top}\boldsymbol{R}_{\text{cct}})\|$

$e_{\text{cct}} = r_{\text{err,cct}} + \xi_{\text{err,cct}}$

if $e_{cct} < c_{\text{act}}(\hat{r}_{\text{act}} + \hat{\xi}_{\text{act}})$ and $\|\boldsymbol{r}_{\text{tar}} - \boldsymbol{r}_{\text{cur}}\| < r_{\text{thrd}}$ **then**

$$\begin{cases} \dot{\boldsymbol{r}}_{\text{des}} = \boldsymbol{r}_{\text{cct}}^{+} - \boldsymbol{r}_{\text{cur}} \\ \dot{\boldsymbol{\xi}}_{\text{des}} = \log(\boldsymbol{R}_{\text{cur}}^{\top}\boldsymbol{R}_{\text{cct}}^{+}) \end{cases}$$

else

$$\begin{cases} \dot{\boldsymbol{r}}_{\text{des}} = \boldsymbol{f}(\boldsymbol{r}_{\text{cur}}) = -(\boldsymbol{r}_{\text{cur}} - \boldsymbol{r}_{\text{via}}) \\ \dot{\boldsymbol{\xi}}_{\text{des}} = \log(\boldsymbol{R}_{\text{cur}}^{\top}\boldsymbol{R}_{\text{via}}) \end{cases}$$

end if

$$\begin{cases} \boldsymbol{r}_{\text{des}} = \boldsymbol{r}_{\text{cur}} + k\Delta t\, \dot{\boldsymbol{r}}_{\text{des}} \\ \boldsymbol{R}_{\text{des}} = \boldsymbol{R}_{\text{cur}} \exp(k\Delta t\, \dot{\boldsymbol{\xi}}_{\text{des}}) \end{cases}$$

The relative next state predicted by the GPR at the estimated via-point is

$$\boldsymbol{p}_{\text{gpr,via}}^{\text{rel}} = GPR\Big(\|\boldsymbol{r}_{\text{via}} - \boldsymbol{r}_{\text{tar}}\|, \|\log(\boldsymbol{R}_{\text{via}}^{\top}\boldsymbol{R}_{\text{tar}})\|\Big), \quad (8)$$

with the global-frame state $\boldsymbol{p}_{\text{gpr,via}} = T(\boldsymbol{p}_{\text{gpr,via}}^{\text{rel}})$. The relative deviations are

$$\tilde{\boldsymbol{r}}_{\text{gpr,via}} = \boldsymbol{r}_{\text{gpr,via}} - \boldsymbol{r}_{\text{via}}, \quad \tilde{\boldsymbol{\xi}}_{\text{gpr,via}} = \log(\boldsymbol{R}_{\text{via}}^{\top}\boldsymbol{R}_{\text{gpr,via}}). \quad (9)$$

Activation thresholds for the GPR are defined as

$$\hat{r}_{\text{act}} = \|\tilde{\boldsymbol{r}}_{\text{gpr,via}}\|, \quad \hat{\xi}_{\text{act}} = \|\tilde{\boldsymbol{\xi}}_{\text{gpr,via}}\|, \quad r_{\text{thrd}} = \max_{i}(\|\boldsymbol{r}_{\text{cc},0,i}\|), \quad (10)$$

Here, $\max_{i}(\|\boldsymbol{r}_{\text{cc},0,i}\|)$ is the maximum among the first points of each CC in $\{\boldsymbol{CC}\}_i$.

When the relative errors are defined as $\tilde{\boldsymbol{r}} = \boldsymbol{r}_{\text{tar}} - \boldsymbol{r}_{\text{cur}}$ and $\tilde{\boldsymbol{\xi}} = \log(\boldsymbol{R}_{\text{cur}}^{\top}\boldsymbol{R}_{\text{tar}})$, then the desired velocity at next step for successful task execution derived from the GPR model is formulated as

$$\begin{cases} \dot{\boldsymbol{r}}_{\text{des}} = \boldsymbol{r}_{\text{cct}}^{+} - \boldsymbol{r}_{\text{cur}} & (11a) \\ \dot{\boldsymbol{\xi}}_{\text{des}} = \log(\boldsymbol{R}_{\text{cur}}^{\top}\boldsymbol{R}_{\text{cct}}^{+}). & (11b) \end{cases}$$

The state $\boldsymbol{p}_{\text{cct}}^{+} = [\boldsymbol{r}_{\text{cct}}^{+} \;\; \boldsymbol{\xi}_{\text{cct}}^{+}]^{\top}$ is the desired state for the next time step and can be obtained from the following

$$\boldsymbol{p}_{\text{cct}}^{\text{rel},+} = GPR(\|\boldsymbol{r}^{+} - \boldsymbol{r}_{\text{tar}}\|, \|\log(\boldsymbol{R}^{+^{\top}}\boldsymbol{R}_{\text{tar}})\|), \quad (12)$$

with $\boldsymbol{p}_{\text{cct}}^{+} = T(\boldsymbol{p}_{\text{cct}}^{\text{rel},+})$, and the target state $\boldsymbol{p}_{\text{tar}}$ can be arbitrarily specified without requiring a demonstrated target, enabling generalization to unseen targets.

The state $\boldsymbol{p}^{+} = [\boldsymbol{r}^{+} \;\; \boldsymbol{\xi}^{+}]^{\top}$ is defined as

$$\begin{cases} \boldsymbol{r}^{+} = \boldsymbol{r}_{\text{cur}} + \alpha_r \Delta r_{\max} \frac{\tilde{\boldsymbol{r}}}{\|\tilde{\boldsymbol{r}}\|}, & (13a) \\ \boldsymbol{\xi}^{+} = \log(\boldsymbol{R}^{+}), & (13b) \\ \boldsymbol{R}^{+} = \boldsymbol{R}_{\text{cur}} \exp\Big(\alpha_\xi \Delta\xi_{\max} \frac{\tilde{\boldsymbol{\xi}}}{\|\tilde{\boldsymbol{\xi}}\|}\Big), & (13c) \end{cases}$$

where the scaling factors α_r and α_ξ are given by

$$\alpha_r = \min\left(\frac{\|\tilde{\boldsymbol{r}}\|}{\Delta r_{\max}}, 1\right), \quad \alpha_\xi = \min\left(\frac{\|\tilde{\boldsymbol{\xi}}\|}{\Delta\xi_{\max}}, 1\right). \quad (14)$$

The values $\Delta r_{\max}$ and $\Delta\xi_{\max}$ specify the allowable translational and rotational step sizes at each update, which serve as safety limits during execution.

Finally, the desired velocity for the next step is determined subject to the GPR activation condition, as summarized in Algorithm 1 with selected constant c_{act}. The constant c_{act} is selected within $0 < c_{\text{act}} \leq 1$ to preserve its physical meaning as a criterion indicating whether the current robot state is within a feasible neighborhood of the motion constraint state.

III. Experiment

A. Experimental setup

The proposed constraint-aware dynamical system framework, which captures task-specific variations such as motion constraints, was evaluated on a 7-dof collaborative manipulator, Franka Emika Panda (Franka Robotics Co.). The experimental setup is shown in Fig. 4 (a).

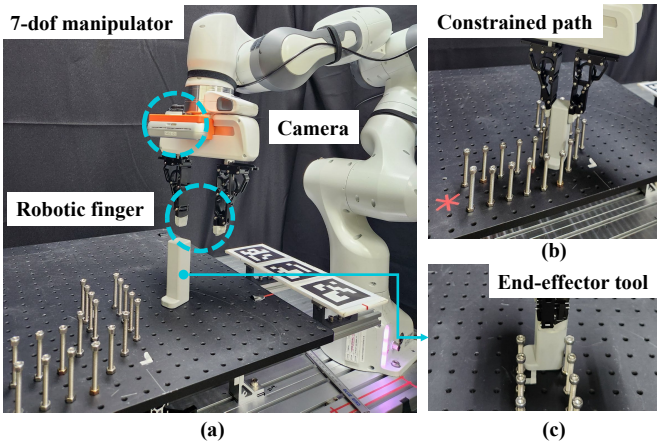

Fig. 4. Experimental setup. (a) Real-robot manipulator with passive gripper (b) Constrained path with bolts wall. (c) Customized tool-tip held by end effector.

A variable-impedance joint PD controller was employed to compute the desired joint torque:

$$\begin{cases} \boldsymbol{\tau}_c = (1 - \gamma_h(\boldsymbol{\tau}_{\text{ext}}))(\boldsymbol{P}\tilde{\boldsymbol{q}} + \boldsymbol{D}\dot{\tilde{\boldsymbol{q}}}) & (15a) \\ \gamma_h(\boldsymbol{\tau}_{\text{ext}}) = 0.5\left(1 + \tanh\left(\frac{\max(\|\boldsymbol{\tau}_{\text{ext}}\|)}{\beta\gamma_{\min}} - \delta\right)\right) & (15b) \end{cases}$$

$$\begin{cases} \boldsymbol{P} = [1500, 1500, 1500, 1500, 170, 100, 50]^{\top} & (16a) \\ \boldsymbol{D} = [60, 60, 60, 60, 25, 20, 10]^{\top} & (16b) \end{cases}$$

Here, $\gamma_h(\boldsymbol{\tau}_{\text{ext}})$ modulates compliance based on the external wrench $\boldsymbol{\tau}_{\text{ext}}$, while $\tilde{\boldsymbol{q}}$ and $\dot{\tilde{\boldsymbol{q}}}$ denote joint position and velocity errors. The parameter $\gamma_{\min}$ sets the minimum external input for adaptation, and δ balances tracking accuracy with compliance.

The desired velocity and state in task space were first computed from the proposed framework and then converted

to joint-space velocities using the Jacobian inverse for inverse kinematics. These joint-space velocities were subsequently integrated to obtain the desired joint positions for control.

For the robotic gripper, we employed the OMEGA GRIPPER (March Bionics Co.), a fully passive robotic finger [17], which does not require additional computation when operating in proximity to the task surface.

B. Experiments and results

Demonstration data were collected via kinesthetic teaching by recording the end-effector state trajectory $\boldsymbol{p} \in \mathbb{R}^3 \times \mathfrak{so}(3)$. In our task, the robot first grasps a designated tool and then performs a constrained path-following motion; accordingly, the grasped tool trajectory was recorded in the same representation. For reliable perception, SAM2 [18] was used for segmentation and FoundationPose [19] for tool pose estimation. In the experiment, the robot grasps the tool from a board and follows a path defined by long bolts, highlighting the coupling between grasp pose accuracy and trajectory reproduction fidelity. Adherence to the learned position–orientation motion constraints is essential for successful task execution and for preventing tool jamming along the path (Fig. 4 (b), (c)). After a successful grasp, the robot proceeds with the constrained path-following task, which inherently evaluates both the grasped configuration and the ability of the learned DS to enforce motion constraints in position and orientation simultaneously.

We compared four dynamical system (DS) methods: *SEDS* [8], *LPV-DS* [12], the mean-trajectory DS (*MT*) [9], and our proposed constraint-aware DS. For a fair comparison, baseline demonstrations were heuristically trimmed to resemble the CC data extracted by our method, thereby aligning trajectories with the constrained path while excluding the pre-grasp phase. In contrast, the proposed method directly exploited the full demonstrations, from grasp to task completion, without heuristic preprocessing.

- **SEDS:** State $[\boldsymbol{r}, \boldsymbol{\xi}]^\top$; five demonstrations heuristically selected.
- **LPV-DS:** 2D positional state $[r_x, r_y]^\top$; higher-dimensional optimization was unstable, and a cylindrical tooltip was used.
- **MT:** State $[\boldsymbol{r}, \boldsymbol{\xi}]^\top$; three representative demonstrations selected heuristically.
- **Proposed:** Six full demonstrations (grasp + constrained execution), naturally encoding position–orientation consistency.

For efficient training, the SVHGP model [20] was adopted as the GPR model and implemented using `GPflow` [21]. For executing highly constrained tasks with the proposed DS, the parameter c_{act} in Algorithm 1 was determined empirically.

The demonstrated trajectories are illustrated in Fig. 5. Fig. 5 (a) shows the complete demonstrations, including both pre-grasp and post-grasp phases, while Fig. 5 (b) presents the selected GC data after grasping, represented in the target frame. The *robot grasp-point* denotes the demonstrated grasping configuration of the robot, the *grasped object position* corresponds to the vision-tracked tool position, and the *estimated grasp-point* is inferred by the proposed method without assuming a predefined grasping location. The estimated grasp-point closely approximates the demonstrated grasp-point.

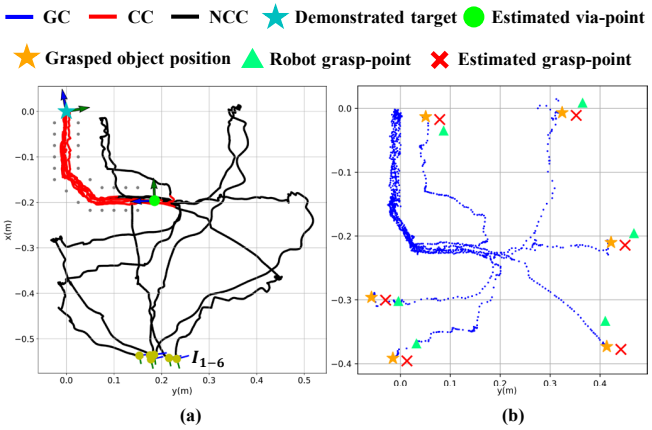

Fig. 5. Demonstrated trajectories and grasping points from $N = 6$ demonstrations. (a) Complete trajectories, including pre- and post-grasp phases, plotted in the target frame. (b) Selected GC data in the target frame, showing the demonstrated robot grasp-points estimated from robot configurations, the vision-tracked object positions, and the grasp-points estimated by the proposed method. A constant two-dimensional shift is applied to the vision-tracked object trajectories for visualization only to compensate for systematic perception bias. Despite this bias, the estimated grasp-point lies within a feasible region of the demonstrated robot grasp-point. CC data and the via-point for task execution are also visualized.

When the vision-logged object trajectories and the logged robot end-effector configurations are independently plotted relative to the same target frame, a small but consistent discrepancy is observed due to systematic perception error during demonstration logging. For visualization only, a constant two-dimensional offset corresponding to this bias is applied to the vision-tracked object trajectories in Fig. 5 (b) to align the plotted quantities in a common frame.

During task execution, we assume the perception error does not differ significantly from that observed during demonstration logging. Despite this bias, the estimated grasp-point lies within a feasible region of the demonstrated grasp-point, confirming the robustness of the proposed estimation process against moderate systematic perception errors. Additionally, the CC data used to model motion constraints, as well as the via-point for initiating task execution, are also visualized in the figure.

Each experiment was repeated 20 times from different initial configurations to evaluate the constrained path-following task. For a fair comparison, the experimented configurations included the estimated via-point as the initial state. The comparative results are summarized in Table III and visualized in Fig. 6. Here, the smoothness is computed as $1/(1+\bar{S})$, where $\bar{S}$ is the time-normalized mean squared acceleration. Since the critical configuration region of the task can be modeled, obstacles can be defined to incorporate these constraint regions, thereby enabling obstacle avoidance and further improving the success rate of the proposed method (see Fig. 6 (d), *Proposed*). Please refer to the supplementary video for detailed experimental results.

TABLE III

THE SUCCESS RATE OF EXECUTING CONSTRAINED PATH-FOLLOWING

Method	Success / Total	Success rate	Smoothness
SEDS	0/20	**0**	0.656 ± 0.290
LPV-DS	0/20	**0**	0.603 ± 0.216
MT	11/20	**0.55**	0.380 ± 0.201
Proposed	20/20	**1**	0.514 ± 0.048

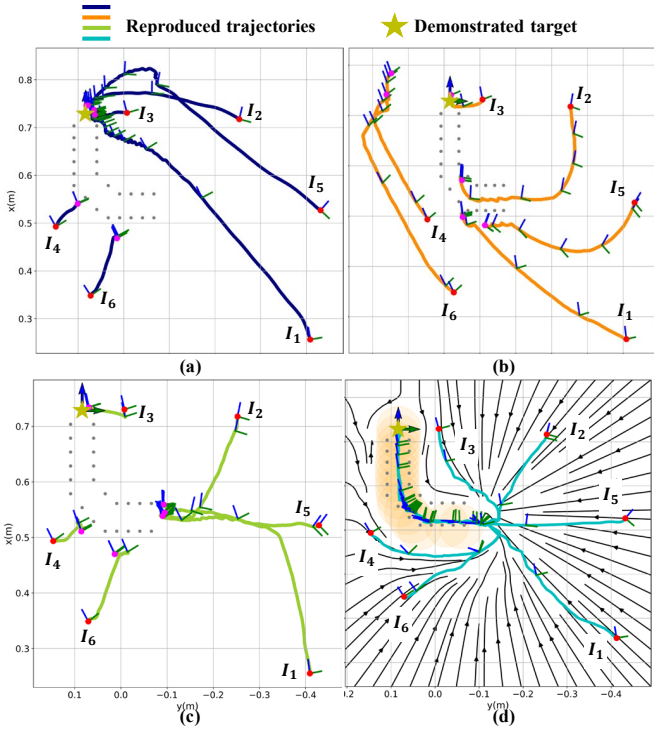

Fig. 6. Reproduced trajectories (post-grasp) for each compared method: (a) *SEDS*, (b) *LPV-DS*, (c) *MT*, and (d) *Proposed*. Each trajectory was initialized from random initial configurations, denoted by I_i. The proposed method achieved the highest task success rate by reliably reproducing the demonstrated orientation states. In addition, incorporating obstacles representing the region of motion constraints further enhanced the success rate by providing obstacle avoidance.

C. Scenario : Dishware collection

The applicability of the proposed constraint-aware dynamical system to real-world scenarios was demonstrated through a dishware collection task, as illustrated in Fig. 7. Dishware collection encompasses various examples of constrained path-following in daily life, such as stacking cups, placing plates into a rack, or inserting utensils into holders.

For autonomous dishware collection, the proposed LfD-based dynamical system was integrated with a VLM–LLM task planner. The planner infers tasks and identifies task-relevant objects in previously unseen scenes without manual annotations, unlike [22]. A VLM (GPT-4.0) analyzes sampled demonstration frames to determine the associated objects, whose regions are then extracted using Grounding DINO [23] and SAM [24]. These regions are passed to FoundationPose [19] to estimate their poses relative to the demonstrations.

Within the integrated framework, scene observations and

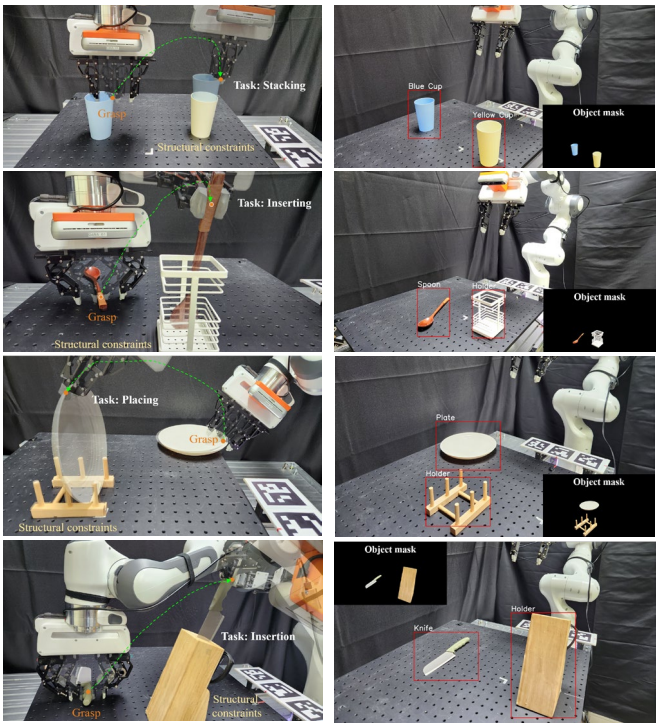

Fig. 7. Dishware collection scenario. (a) The robot acquires highly constrained daily-life tasks through human demonstrations. (b) The learned skills are executed in a new scene via VLM-based task planning.

natural language prompts are processed by the VLM–LLM planner to generate task sequences. Unlike conventional approaches relying on motion planners that require repeated optimization and feasibility checks, the proposed LfD-based dynamical system directly generates constraint-consistent motions from demonstrations to execute the planned sequences.

Four types of dishware were used in the experiments. After learning the constraints from demonstrations, the robot autonomously performed the tasks in novel scenes, including cases where the tool is located in a different configuration from the ones demonstrated. The system consistently succeeded even under human disturbances, as shown in the supplementary video.

IV. DISCUSSION

The proposed approach demonstrated superior performance in a highly constrained task by explicitly modeling motion constraints. Nevertheless, several limitations remain.

Since the motion constraints are learned via Gaussian process regression (GPR), the accuracy of the model depends on the target configuration inputs, $\|\boldsymbol{r} - \boldsymbol{r}_{\mathrm{tar}}\|$ and $\|\log(\boldsymbol{R}^{\top}\boldsymbol{R}_{\mathrm{tar}})\|$. In cases where a monotonic approach to the target along the critical configuration routes (CCs) cannot be guaranteed, heuristic tuning of GPR hyperparameters (e.g., kernel length scale and variance) may be required. In our experiments, both were set to 1.0 and generally performed well, though occasional adjustment and model verification may still be necessary.

In addition, the proposed dynamical system introduces the variable c_{act}, as described in Algorithm 1. This value is not fixed and may vary across experimental trials for different tasks. Similarly, the threshold parameter d_{thrd} must be carefully chosen below the bound d_{bound}, which introduces sensitivity to hyperparameter selection. Therefore, achieving automated parameter tuning remains an important direction for future work.

Despite these limitations, the Lie algebra-based formulation offers a promising direction for extending the method beyond trajectory-level constraints. In particular, it could incorporate wrench-related constraints, enabling applications at both force/wrench and trajectory levels. Future work will address the current limitations while exploring such extensions to broaden the applicability of the proposed method.

V. Conclusion

This paper presents a novel framework for generating a constraint-aware autonomous dynamical system (DS) capable of executing highly constrained tasks by learning from human demonstrations. A core contribution of our approach is the automatic extraction of task-specific constraints ensuring task success from demonstration data, allowing motion generation without prior knowledge of the task.

The modeling of task-specific constraints is conducted in three stages. First, graspable configuration (GC) data — the relative configurations between tool and robot observed during successful task execution, where the robot holds the tool stably—are selected to define the *grasp-point* for the target task, specifying where the robot grips the task tool. Next, *via-point*, indicating where motion constraint adherence begins, is estimated from selected critical configuration (CC) data, which consist of trajectory points consistently observed across successful demonstrations, using thresholds derived from failed demonstrations. Finally, *task-specific motion constraints* are captured by modeling the critical configuration region and trajectory (CCR and CCT) using Gaussian Process Regression (GPR), representing the probabilistic structure of successful motions.

Experimental results demonstrate that the proposed framework improves task success rates compared to other *state-of-the-art* methods in Table I, confirming its effectiveness in real-world scenarios such as dishware collection.

The framework is expected to be applicable to a variety of scenarios, including human–robot interaction tasks that require robust execution in the face of unexpected perturbations and mobile robot navigation in constrained environments, where task-specific constraints guide safe and efficient motion.

References

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

[2] K. Qian, X. Xu, H. Liu, J. Bai, and S. Luo, "Environment-adaptive learning from demonstration for proactive assistance in human–robot collaborative tasks," *Robotics and Autonomous Systems*, vol. 151, p. 104046, 2022.

[3] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 339–354.

[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," *Advances in neural information processing systems*, vol. 26, 2013.

[6] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[7] A. Duan, I. Batzianoulis, R. Camoriano, L. Rosasco, D. Pucci, and A. Billard, "A structured prediction approach for robot imitation learning," *The International Journal of Robotics Research*, vol. 43, no. 2, pp. 113–133, 2024.

[8] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[9] S. Jadav, J. Heidersberger, C. Ott, and D. Lee, "Shared autonomy via variable impedance control and virtual potential fields for encoding human demonstration," *arXiv preprint arXiv:2403.12720*, 2024.

[10] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.

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

[12] N. B. Figueroa Fernandez and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," *Proceedings of Machine Learning Research*, 2018.

[13] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 472–477.

[14] A. Shukla and A. Billard, "Coupled dynamical system based arm–hand grasping model for learning fast adaptation strategies," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 424–440, 2012.

[15] M. Khoramshahi and A. Billard, "A dynamical system approach to task-adaptation in physical human–robot interaction," *Autonomous Robots*, vol. 43, pp. 927–946, 2019.

[16] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, 2015.

[17] D. Yoon and K. Kim, "Fully passive robotic finger for human-inspired adaptive grasping in environmental constraints," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 3841–3852, 2022.

[18] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[19] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.

[20] H. Liu, Y.-S. Ong, and J. Cai, "Large-scale heteroscedastic regression via gaussian process," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 708–721, 2020.

[21] A. G. d. G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. Le, Z. Ghahramani, J. Hensman *et al.*, "Gpflow: A gaussian process library using tensorflow," vol. 18, no. 40, pp. 1–6, 2017.

[22] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," in *Conference on Robot Learning*. PMLR, 2023, pp. 3418–3433.

[23] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.

[24] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.