A comparative study of skill learning for task composition in human-robot collaborative scenarios

A comparative study of skill learning for task composition in human-robot collaborative scenarios

Cesare Luigi Blengini, Rosario Francesco Cavelli, Pangcheng David Cen Cheng, Marina Indri, Carlo Migliaccio
Dipartimento di Elettronica e Telecomunicazioni - Politecnico di Torino
Corso Duca degli Abruzzi, 24, 10129, Torino, Italy
{cesare.blengini, rosario.cavelli, pangcheng.cencheng, marina.indri}@polito.it, carlo.migliaccio@studenti.polito.it

***Abstract*—Learning from Demonstration (LfD) paradigm has been widely developed in the last decade in the context of Human-Robot Collaboration (HRC). It allows for the collection of demonstrations from several types of sources and the teaching of specific motions or trajectories to the robot that could be difficult to be implemented with traditional methods. In this paper, we present an LfD pipeline that gathers data from a human expert and teaches the robot low-level skills, so it can perform higher-level tasks by means of task composition. Three learning methods are compared: Goal-Conditioned Behavioral Cloning, Dynamic Movement Primitives, and Gaussian Mixture Model with Regression. The low-level skills that have been trained are PICK, PLACE, and POUR. The approach has been tested in a real cobot with an RGB-D camera and ArUCO markers, which are used for easier skill parametrization and generalization.**

***Index Terms*—Learning from Demonstration, Human-Robot Collaboration, Task decomposition**

I. Introduction

Human-robot collaboration (HRC) has become a standard in scenarios where robotic systems are expected to be flexible, rapidly reconfigurable, and easy to program. In particular, the need to quickly program the manipulator for customized motions in many applications led to the exploration of alternative approaches to path planning. Customized paths are often difficult to program using classical approaches; however, in an HRC context with collaborative manipulators (cobots), there are distinct paradigms for learning skills from demonstrations [1]. In this context, Learning from Demonstration (LfD) is a promising paradigm, because it enables the acquisition of robot behaviors from examples, e.g., manually guiding the robot's joints in different positions and training a model to learn specific movements. In this way, it is possible to reduce the dependence on manual programming and task-specific controller design [2].

This perspective naturally suggests a hierarchical view of manipulation, hence of skill learning. Complex tasks can be decomposed into elementary actions, which can be interpreted as low-level skills. The main advantage of demonstrating sub-tasks is that the human teacher can provide clear and precise demonstrations for each separate step [3]. For example, the authors in [4] present a framework for robust imitation learning, which divides complex tasks into simpler subtasks from the demonstrated trajectories, saved as skill primitives for flexible reuse and then exploited in skill generalization to use the learned skills in new situations.

This study was carried out within the Space It Up project funded by the Italian Space Agency, ASI, and the Ministry of University and Research, MUR, under contract n. 2024-5-E.0 - CUP n. I53D24000060005.

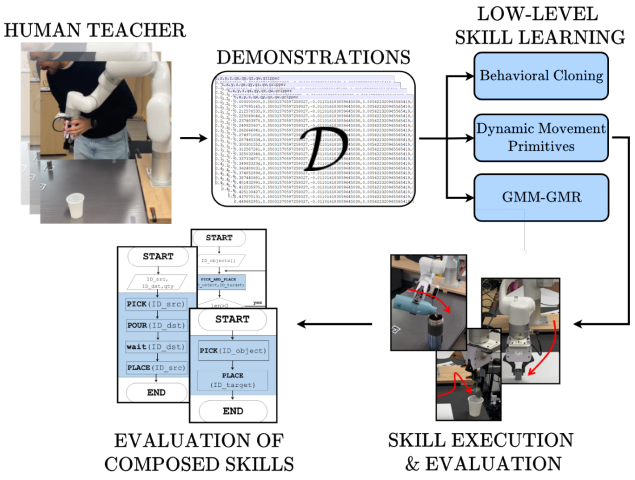


Fig. 1: Three motion encoders (GCBC, DMP, and GMM-GMR) are compared within a unified LfD pipeline on their ability to learn reusable low-level skills that are composed into higher-level tasks.

Demonstrations from humans can be collected online or offline, so the teaching process can also be classified as online/offline teaching with the corresponding counterparts of online/offline learning. The combination of online teaching with offline learning includes approaches like Behavioral Cloning (BC) [5], Dynamic Movement Primitives (DMP) [6], [7], where the teaching methods are based on kinesthetics, teleoperation, sensors, and virtual reality [8].

The choice of the motion encoder is a central issue in low-level skill learning. Gaussian Mixture Models with Gaussian Mixture Regression (GMM-GMR), BC, and DMP are among the most widely adopted approaches [9]–[11]. Nevertheless, they rely on substantially different representations of the demonstrated behavior, and recent surveys on LfD [12], [13] and on its practical deployment in manufacturing [3] highlight that this choice strongly affects accuracy, smoothness, and goal compliance. However, no consensus has been reached on which formulation should be preferred when skills are intended to be composed into higher-level tasks.

Alternative approaches, such as the one in [14], for imitation learning using demonstrations are presented. The learning pipeline is deployed in a UR5 robot, where the understanding process from the demonstrations translates the captions of

the videos into commands, and the commands are executed using a manipulation module with instance segmentation and affordance manipulation prediction models. Moreover, in [15], the authors presented a hierarchical reinforcement learning framework to perform complex manipulation tasks. It combines human prior knowledge by using a Policy Gradient with a Parameter-based exploration (PGPE) method. However, the approach is constrained to sim-to-real transfer methods and only validated for peg-in-hole applications.

In this paper, a low-level skill is intended as a reusable motion primitive, learned from demonstration and later invoked to build a higher-level task. Under this definition, the quality of a learned skill cannot be evaluated only in terms of trajectory imitation, but must also be discussed in terms of generalization, task-dependent adaptation, and suitability for subsequent composition. Preliminary results of the proposed LfD pipeline are available in [16]. Also, the BC, DMP, and GMM-GMR models are compared not only on isolated skill reproduction, but also on their ability to be composed into higher-level tasks, with the aim of identifying the most suitable encoder for task composition (Figure 1). The study provides a preliminary but structured experimental basis to support future research on skill reuse and hierarchical task composition for collaborative industrial manipulation scenarios. To isolate the effect of the representation, the comparison is carried out within a unified LfD pipeline, in which data collection, operational-space skill representation, task-parameter retrieval, and trajectory execution are kept fixed, while only the motion encoder varies. Moreover, to evaluate the LfD pipeline under realistic conditions, the models are tested with few demonstrations per skill. In practice, kinesthetic demonstrations from a single human demonstrator provide limited data, making the large datasets typically required for deep learning impractical.

The remainder of the paper is organized as follows. Section II describes the proposed LfD pipeline, while Section III illustrates the experimental setup and discusses the obtained results. Finally, Section IV concludes the paper, highlighting its current limitations and future improvements.

II. Proposed Approach

The proposed approach consists of an LfD pipeline designed to compare different frameworks for low-level robotic skill learning, to evaluate how they behave when the learned skills are composed into more complex manipulation tasks.

As summarized in Figure 2, the pipeline is organized in four stages: *demonstration collection*, *data pre-processing*, *motion encoding (learning)*, and *trajectory execution (inference and reproduction)*. Data acquisition and the common pre-processing steps are performed only once and feed all encoders. In this way, any observed performance gap can be ascribed to the learning approach rather than to differences in the upstream data.

A. Collection of demonstrations

Demonstrations are acquired through kinesthetic teaching: the operator, who is assumed to work with a collaborative manipulator, physically guides the robot along the desired trajectory while the robot's state is recorded. In particular, the Tool Center Point (TCP) position and orientation parameters are recorded, alongside the state of the gripper.

This allows to work entirely in task space, avoiding the correspondence problem typical of joint-space and teleoperated demonstrations [12], [17]. Furthermore, it yields learning skills that are independent of the kinematic structure of the robot, and hence reusable on different platforms. This robot-agnostic representation also strengthens the purpose of the LfD pipeline and the performance comparison of the different motion encoders.

Each demonstration d is a sequence of time-stamped TCP poses sampled at a fixed rate:

$$d = \left\{ t,\ x,\ y,\ z,\ q_x,\ q_y,\ q_z,\ q_w,\ p_{grip} \right\}_{t=0}^{T_{max}}, \tag{1}$$

where x, y, z are the Cartesian coordinates of the TCP, q_x, q_y, q_z, q_w are the components of the TCP unit quaternion orientation, p_{grip} is the gripper opening, and T_{max} is the demonstration duration. For each low-level skill, K demonstrations are recorded and collected into a dataset $\mathcal{D} = \{d_k\}_{k=1}^{K}$.

B. Common data pre-processing

The data coming from each recorded demonstration is pre-processed by applying a sequence of steps that are commonly used for trajectories on $SE(3)$.

Temporal resampling and interpolation: Since human-driven demonstrations inevitably vary in duration, a temporal alignment step is required. Each trajectory is resampled onto a uniform time grid $\mathbf{t}_{grid}$ with constant timestep dt. To obtain the missing data, Cartesian components (x, y, z) are linearly interpolated, while the quaternion components are interpolated through Spherical Linear Interpolation (SLERP).

Quaternion normalization: Each quaternion sample $\mathbf{q}(t) = [q_x,\ q_y,\ q_z,\ q_w]$ is enforced to satisfy

$$\|\mathbf{q}(t)\| = 1, \tag{2}$$

in order to compensate for the numerical drift accumulated by finite-precision sensor measurements and by the interpolation step, thus keeping the rotation representation valid.

Hemisphere continuity: Since $\mathbf{q}$ and $-\mathbf{q}$ encode the same rotation, unchecked sign flips between consecutive samples can introduce spurious π-radian jumps. Continuity is therefore enforced by requiring

$$\mathbf{q}(t) \cdot \mathbf{q}(t+1) > 0, \tag{3}$$

and inverting the sign of $\mathbf{q}(t+1)$ whenever the dot product is negative.

Quaternion-to-angle-axis mapping: Since the DMP and GMM-GMR motion encoders operate in Euclidean vector spaces, orientation quaternions are remapped to the angle-axis representation through the quaternion logarithm,

$$\mathbf{r}(t) = \log(\mathbf{q}(t)) = [r_x,\ r_y,\ r_z], \tag{4}$$

yielding a minimal three-dimensional parameterization that is locally Euclidean, and therefore compatible with linear

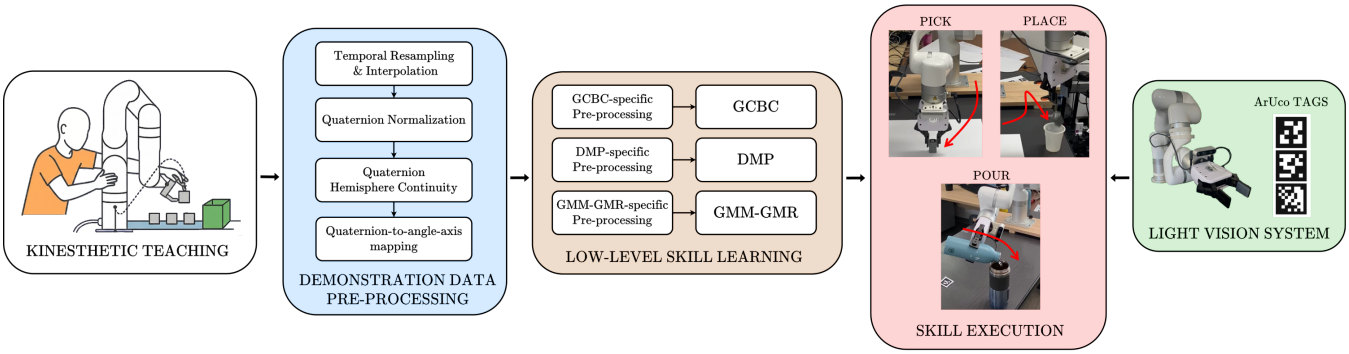


Fig. 2: Overview of the proposed LfD pipeline. Kinesthetic demonstrations undergo a shared pre-processing stage and feed three alternative motion encoders (GCBC, DMP, GMM-GMR), whose learned skills are executed on a UFACTORY xArm6 with task parameters retrieved online by an ArUco-based vision system.

operations such as finite differencing, weighted averaging, and basis-function evaluation.

C. Goal-Conditioned Behavioral Cloning (GCBC)

The GCBC approach [18], [19] extends the standard BC by formulating the skill learning as a supervised regression problem in which the policy is explicitly conditioned on a goal. The model approximates a policy π_θ that, given the current task-space state of the end-effector $\mathbf{s}_t \in \mathcal{S}$ and a goal $\mathbf{g} \in \mathcal{G}$ specifying the desired target configuration, predicts the corresponding action $\mathbf{a}_t \in \mathcal{A}$ required to reproduce the demonstrated behavior toward $\mathbf{g}$.

Within the proposed pipeline, the state $\mathbf{s}_t = [\mathbf{p}_t,\ \mathbf{r}_t] \in \mathbb{R}^6$ collects TCP position $\mathbf{p}_t$ and angle-axis orientation $\mathbf{r}_t$, and the goal $\mathbf{g} = [\mathbf{p}_g,\ \mathbf{r}_g] \in \mathbb{R}^6$ is represented in the same task-space form as the state, encoding the desired final TCP pose. The action $\mathbf{a}_t = [\Delta\mathbf{p}_t,\ \Delta\mathbf{r}_t] \in \mathbb{R}^6$ is the one-step finite difference of the state. In this way, the policy effectively predicts incremental motions rather than absolute poses, conditioned on the target to be reached. The state-goal-action triplets from all K demonstrations are stacked into matrices $\mathbf{I} \in \mathbb{R}^{M\times 12}$ and $\mathbf{O} \in \mathbb{R}^{M\times 6}$, representing the input (concatenation of state and goal) and the desired output of the regression, respectively. Following the standard GCBC formulation, the goal associated with each demonstration is taken as the final state of the corresponding trajectory. As a model-specific processing step, such matrices are normalized by applying per-column z-score normalization:

$$\mathbf{I}_n = \frac{\mathbf{I} - \mathbf{i}_{\mathbf{mean}}}{\mathbf{i}_{\mathbf{std}}}, \quad \mathbf{O}_n = \frac{\mathbf{O} - \mathbf{o}_{\mathbf{mean}}}{\mathbf{o}_{\mathbf{std}}}, \tag{5}$$

where $\mathbf{i}_{\mathbf{mean}}, \mathbf{o}_{\mathbf{mean}}, \mathbf{i}_{\mathbf{std}}$ and $\mathbf{o}_{\mathbf{std}}$ are the mean and the standard deviation of the input and output matrices, respectively. The normalization parameters are stored alongside the model and reused at inference.

The policy $\pi_\theta(\mathbf{s}_t, \mathbf{g})$ is encoded as a Multi-Layer Perceptron (MLP) with four hidden layers of size $l_i \in [32, 256]$, Rectified Linear Unit (ReLU) or $tanh$ activations, and trained adopting the Mean Squared Error (MSE) loss function. For motion generation, an iterative rollout procedure from the initial state $\mathbf{s}_0$ toward a user-specified goal $\mathbf{g}$ is exploited: at each step, the robot's state $\mathbf{s}_t$ is concatenated with $\mathbf{g}$ and normalized using $\mathbf{i}_{\mathbf{mean}}$, $\mathbf{i}_{\mathbf{std}}$, $\mathbf{o}_{\mathbf{mean}}$ and $\mathbf{o}_{\mathbf{std}}$, then fed to the MLP to compute $\mathbf{a}_t$ through forward pass, whose output is then denormalized, and the resulting $(\Delta\mathbf{p}_t, \Delta\mathbf{r}_t)$ are applied to $\mathbf{s}_t$ to obtain $\mathbf{s}_{t+1}$. The goal $\mathbf{g}$ remains fixed throughout the rollout. The angle-axis representation is remapped to unit quaternion space through the exponential map; once combined with the TCP positions at each time step, they produce the generalized end-effector trajectory.

D. Dynamic Movement Primitives (DMP)

The DMP model encodes each demonstrated skill as a nonlinear dynamical system formed by a critically damped attractor toward the goal. The characteristics of the demonstrated trajectory are enclosed in a forcing term that regulates the behavior of the nonlinear attractor. For each dimension ϕ of the robot's state, the DMP couples the transformation system

$$\tau\dot{v} = \alpha_z(\beta_z(g - \phi) - v) + f(s), \qquad \tau\dot{\phi} = v, \tag{6}$$

with a canonical system $\tau\dot{s} = -\alpha_s s$ that drives the phase variable s from 1 to 0 along the motion, replacing explicit time dependence and ensuring that the forcing term vanishes at convergence. The description of the variables of (6) can be found in [10]. As a method-specific pre-processing step, the velocity $\dot{\phi}(t)$ and the acceleration $\ddot{\phi}(t)$ of each trajectory are obtained via finite differences of the demonstration samples, since the DMP fitting equations require the first two time derivatives of the trajectories. Setting $\beta_z = \alpha_z/4$ enforces critical damping, so that asymptotic convergence to the goal g is guaranteed by construction. The shape of the demonstrated motion is encoded by $f(s)$ as a phase-modulated weighted sum of N Gaussian basis functions, whose weights are obtained by a closed-form weighted linear regression on the target forcing term reconstructed from (6) and from the finite-difference derivatives of the demonstration. The full canonical/transformation system decomposition, the choice of Gaussian basis functions, and the closed-form weight estimation procedure are extensively discussed in [6], [10].

Since the robot state is composed of six components describing the TCP pose, the classical 1D DMP formulation

needs to be extended to six dimensions. Such a 6D model formulation can be obtained by considering an independent 1D formulation for each dimension of the robot state.

In its original formulation, a DMP is fit on a single demonstration. To exploit all K available demonstrations of a given low-level skill, an independent DMP is fit to each trajectory, yielding K weight vectors $\{\mathbf{w}^{(k)}\}_{k=1}^{K}$. The weight vector representing the skill is then obtained as the demonstration-wise average

$$\mathbf{w} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}^{(k)}, \tag{7}$$

which is a common strategy to consolidate multiple demonstrations into a single DMP [6].

The resulting weight vector $\mathbf{w}$ is stored together with the demonstration starting and end-points $\boldsymbol{\phi}_0$ and $\boldsymbol{g}$ for later use at inference. The starting and end-points are taken as the average of the corresponding points across the K demonstrations, consistently with (7).

At inference, a forward Euler integration of the canonical and transformation systems generates the trajectory step by step, by evaluating the basis functions and the forcing term at the current phase, computing the desired acceleration, and integrating velocity and position; the angle-axis components are then remapped to quaternion space via the exponential map.

E. Gaussian Mixture Model and Regression (GMM-GMR)

The GMM-GMR probabilistically encodes the skill by modeling the joint distribution of a motion-phase variable and the task-space variables as a Gaussian mixture, and retrieving a generalized trajectory by conditioning the learned distribution on the phase.

As a method-specific pre-processing step, each demonstration is augmented with a normalized phase variable

$$T_i = \frac{t_{grid,i} - t_0}{t_{end} - t_0} \in [0, 1], \tag{8}$$

which decouples the representation from absolute time and allows demonstrations of different durations to be stacked, into a single observation matrix $Z = [T \mid \mathbf{P}] \in \mathbb{R}^{N\times 7}$, with $\mathbf{P} = [x, y, z, r_x, r_y, r_z]$. The joint distribution of phase and task-space variables

$$p(\xi) = \sum_{k=1}^{C}\pi_k\,\mathcal{N}(\xi \mid \mu_k, \Sigma_k), \qquad \xi = [T,\ \mathbf{P}]^\top, \tag{9}$$

is then fitted via Expectation-Maximization, with convergence monitored on the data log-likelihood and the number of components C selected by inspecting the elbow of the log-likelihood as a function of C, so as to balance model expressiveness against overfitting [20]. Together with the learned parameters $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{C}$, the mean demonstration duration T_{mean} is stored for use at inference.

A generalized trajectory is generated by sweeping a query phase $T^* \in [0, 1]$ and computing, in closed form, the conditional expectation $\hat{\mathbf{P}}(T^*) = \mathbb{E}[\mathbf{P} \mid T^*]$ together with its associated covariance, which provides a point-wise uncertainty measure along the motion [20]. The output duration is set as $T_{out} = \gamma\, T_{mean}$, being $\gamma \in [0.5, 2.0]$ a user-defined temporal rescaling factor, and a uniform time grid $t_i = i \cdot dt$ is mapped to the phase domain via $T_i = t_i / T_{out}$. Unlike DMP, the trajectory emerges from the learned distribution rather than from an explicit attractor, and its endpoint is determined by the regression at $T = 1$.

F. Composition of low-level skills into high-level tasks

The encoders previously introduced produce *low-level skills*, such as `PICK`, `PLACE`, and `POUR`, that constitute the elementary building blocks of the proposed framework. To address operations of practical interest, these primitives must be assembled into more complex behaviors. In this study, a *programmatic composition* scheme is adopted, in which a high-level controller invokes the learned skills following a deterministic sequence that may include loops and conditional branches [6].

A high-level task is therefore defined as an ordered sequence of skill invocations

$$\mathcal{T} = \big\langle\, (\sigma_1, \boldsymbol{\theta}_1),\, (\sigma_2, \boldsymbol{\theta}_2),\, \ldots,\, (\sigma_M, \boldsymbol{\theta}_M)\,\big\rangle, \tag{10}$$

where each σ_j is one of the learned skills and $\boldsymbol{\theta}_j = \{\mathbf{s}_0^{(j)}, \mathbf{g}^{(j)}\}$ collects the task-dependent parameters, i.e., the start and goal poses of the corresponding motion. Before each call, the high-level controller refreshes $\boldsymbol{\theta}_j$ from the current scene description, generates the corresponding trajectory through the encoder-specific inference procedure, and dispatches it to the robot's execution layer. This loop provides a minimal yet effective form of *intra-task generalization*, in which the same learned skills can be replayed in slightly different scene configurations without any re-training.

III. Experimental Results

This section presents the experimental validation of the proposed LfD pipeline, comparing the three motion encoders on the low-level motion skill learned and on composed high-level tasks. The section concludes with a human-robot-collaborative case study on mixed-drink preparation.

A. Robotic platform

The manipulator used in the experiments is a UFACTORY xArm6 [21], a 6-DOF collaborative robotic arm equipped with a two-finger parallel gripper used for grasping and object handling. The robot's built-in gravity-compensation mode allows the operator to physically guide the arm along the desired trajectory, enabling kinesthetic demonstrations without any external teleoperation interface or force/torque sensor.

The software stack is built on top of ROS 2 Jazzy and mixes `rclpy` and `rclcpp` nodes together with the xArm Python SDK. The code is available at [22], which also contains the video showcasing the demonstrations and results.

The generalization of the learned skills to previously unseen object configurations is obtained through a lightweight perception layer based on an Intel RealSense D435 RGB-D camera mounted on the robot wrist in an *eye-in-hand* configuration.

Such a vision system is queried only at inference, to estimate in a deterministic way the poses of task-relevant parameters, i.e., the starting and ending points of each motion. Indeed, the RGB stream is used to detect ArUco fiducial markers [23] positioned on relevant points or objects. An example of the workspace as sensed by the robot through the vision system is shown in Figure 3.

This implementation choice has been made to isolate the effect of the motion encoders: any difference in performance can be attributed to the particular motion encoder, rather than to the different handling points coming from generative algorithms. Furthermore, this strategy enables efficient assessment of the generalization properties of the learned skills to unseen object configurations.

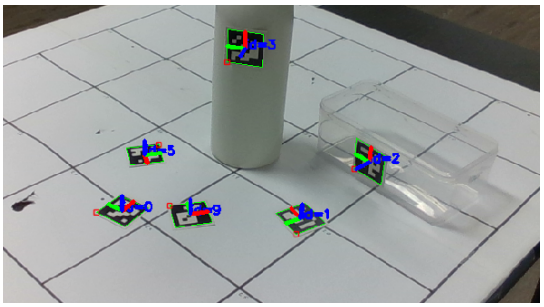


Fig. 3: Output of the ArUco-based vision system used during low-level skill motion generation. For each ArUco tag, the corresponding ID and reference frame are visualized.

B. Demonstrated low-level skills

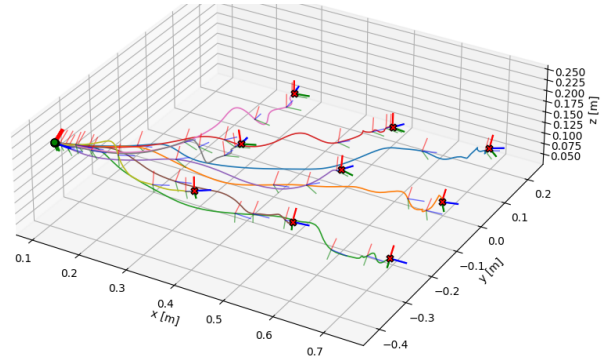
The comparison is carried out on three low-level skills, selected as a minimal but representative set of primitives that can be combined into non-trivial manipulation tasks:

- `PICK` – approach an object identified by its marker and grasp it;
- `PLACE` – transfer the grasped object to a target location (marked with an ArUco tag) and release it;
- `POUR` – tilt a grasped container so as to align its spout with a receiving glass.

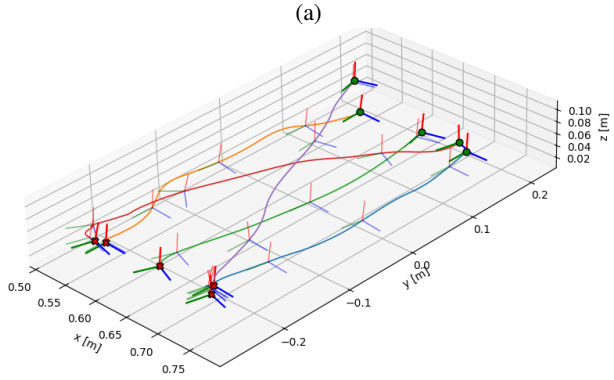
For the three *discrete* primitives (`PICK`, `PLACE`, `POUR`), a dataset of nine, five, and six, respectively, kinesthetic demonstrations has been collected and used to train all the motion encoders introduced in Section II. The collected demonstrations are visualized in Figure 4. As discussed in Section I, only a few demonstrations per skill were collected to reflect realistic working conditions, where repeatedly demonstrating the same skill is impractical.

C. Training hyperparameters

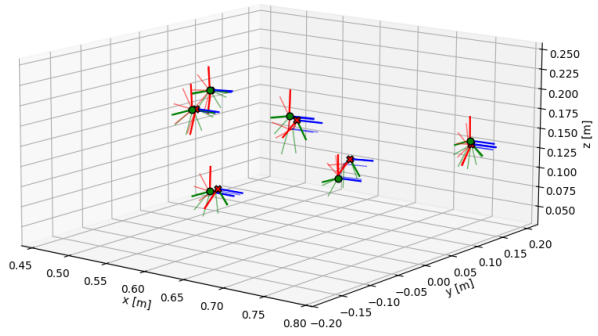
The hyperparameters used for training each motion encoder are summarized in Table I. The specific value actually used for a given encoder-skill pair is selected based on the complexity of the demonstrated motion (length, presence of large orientation changes, number of via-points).



(a)



(b)



(c)

Fig. 4: Trajectories used as demonstrations for learning (a) `PICK`, (b) `PLACE`, (c) `POUR`. For all demonstrations, the starting and ending poses of the TCP are indicated by the green circle and the red cross, respectively. The orientations of the TCP along the trajectories are shown, visualizing its x, y, and z axes in red, green, and blue, respectively.

D. Evaluation metrics

The evaluation relies on a set of metrics targeting three distinct aspects of the implemented LfD pipeline:

- *Trajectory reproduction* of a demonstrated skill under matched conditions,
- *Generalization* of the learned skill to unseen task parameters,
- Success of the *high-level* tasks composed by sequencing the low-level primitives.

TABLE I: Motion-encoders hyperparameters used.

	Hyperparameter	Value
GCBC	MLP layers	4
	Neurons per layer	128 / 128 / 128 / 64
	Activation function	ReLU
	Learning rate	0.001
	Training epochs	4000
	Batch size	128
	Weight decay	0
DMP	Temporal scaling τ	1
	α_z	25
	$\beta_z = \alpha_z/4$	6.25
	α_s	1
	Basis functions N	100
GMM-GMR	Mixture components K	3
	reg_covar	1e-06
	Max iterations of EM algorithm	300
	Convergence tolerance	0.0001

Reproduction metrics. Reproduction fidelity aims to assess whether the dynamics of the demonstrated trajectories were effectively learned by the motion encoders. Such fidelity is evaluated by comparing a generated trajectory with the corresponding demonstrations, under the same start and goal states. To this end, the start and goal TCP poses were taken from the demonstration itself at the first and last time instants. Let $\mathbf{x}^{\text{demo}}(t_i)$ and $\mathbf{x}^{\text{gen}}(t_i)$ be the demonstrated and generated trajectory samples at t_i, respectively, and $\mathbf{x}_T^{(\cdot)}$ their final samples. The reproduction metrics are:

- Root Mean Square Error (RMSE), used to measure the trajectories' similarity at each time instant for N_s number of samples:

$$\text{RMSE} = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}\left\|\mathbf{x}^{\text{demo}}(t_i) - \mathbf{x}^{\text{gen}}(t_i)\right\|^2}. \quad (11)$$

  RMSE for the position and orientation are reported separately as RMSE_{pos} and RMSE_{ori}.
- The Hausdorff distance d_H, that captures time-independent geometric shape similarity:

$$d_H = \max\Big\{ \max_i \min_j \left\|\mathbf{x}^{\text{demo}}(t_i) - \mathbf{x}^{\text{gen}}(t_j)\right\|, \max_j \min_i \left\|\mathbf{x}^{\text{gen}}(t_j) - \mathbf{x}^{\text{demo}}(t_i)\right\| \Big\}, \quad (12)$$

- The motion smoothness, defined as the integrated squared jerk J:

$$J = \int_0^T \|\dddot{\mathbf{x}}(t)\|^2\, dt. \quad (13)$$

Generalization and high-level task metrics. Generalization capabilities of each motion encoder are evaluated for each low-level skill, exploiting the ArUco-based vision layer to retrieve unseen start and goal poses. Since none of the recorded demonstration matches the experimental conditions, it is not possible to exploit the aforementioned metrics. Instead, it is possible to evaluate the success rate of each trained model, defined as:

$$\text{TSR} = \frac{n_{\text{SUCCESS}}}{n_{\text{TOTAL}}}. \quad (14)$$

Similar considerations can also be applied when evaluating the composition of low-level skills into high-level tasks. Besides the TSR of the entire high-level task, in this case, the success rate of each low-level skill composing it has been evaluated as:

$$\text{SSR}_\sigma = \frac{n^\sigma_{\text{SUCCESS}}}{n^\sigma_{\text{TOTAL}}}, \quad (15)$$

with $\sigma \in \{\texttt{PICK}, \texttt{PLACE}, \texttt{POUR}\}$ indicating the low-level skill building the task.

E. Low-level skill reproduction

This section discusses the reproduction performances of the three encoders on `PICK`, `PLACE`, and `POUR`, under matched start and goal conditions . Tables II–IV report, for each metric, the values averaged across the K demonstrations collected for the corresponding skill. The values highlighted in bold correspond to the best obtained results.

TABLE II: `PICK` – reproduction metrics.

	Unit	GCBC	DMP	GMM-GMR
RMSE_{pos}	[m]	0.5374	**0.0546**	0.0865
RMSE_{ori}	[deg]	47.11	10.43	**9.86**
d_H	[m]	0.6567	**0.0532**	0.0667
J	[m/s^3]	4443.21	115.66	**1.92**

TABLE III: `PLACE` – reproduction metrics.

	Unit	GCBC	DMP	GMM-GMR
RMSE_{pos}	[m]	0.0841	0.0851	**0.0679**
RMSE_{ori}	[deg]	6.74	11.76	**6.69**
d_H	[m]	**0.0072**	0.1383	0.0379
J	[m/s^3]	740.60	1127.42	**0.95**

TABLE IV: `POUR` – reproduction metrics.

	Unit	GCBC	DMP	GMM-GMR
RMSE_{pos}	[m]	**0.0026**	0.0030	0.0292
RMSE_{ori}	[deg]	13.17	**6.95**	14.21
d_H	[m]	**0.0009**	0.0048	0.0421
J	[m/s^3]	9.34	4.01	**0.59**

Discussion. The three encoders display complementary behaviors, with no single approach dominating across all skills. GMM-GMR achieves the best smoothness on all three skills, with values of J that are one to three orders of magnitude lower than the competitors. It is also the most accurate on `PLACE` on every reported metric, thanks to its probabilistic encoding that consolidates the variance of the demonstrations into a regular generalized trajectory. DMP dominates the geometric metrics (RMSE_{pos}, d_H) on `PICK` and reproduces the rotation-dominated motion of `POUR` with the lowest RMSE_{ori}, as a direct consequence of the critically damped attractor that anchors the trajectory to the demonstrated goal. GCBC proved to be competitive only on `POUR`, where it attains the best RMSE_{pos} and d_H, indicating that on short and well-localized motions an MLP trained on $(\mathbf{s}_t, \mathbf{g}, \mathbf{a}_t)$ tuples can fit the global geometric mapping accurately; on the longer `PICK` motion, however, the absence of an explicit goal-attractor

mechanism causes the per-step prediction errors to accumulate over the rollout and yields the largest deviations among the three encoders.

F. Generalization capabilities

For each motion encoder, every skill was executed with start and goal poses different from those used during demonstration collection. These poses were defined using ArUco tags in the workspace. A trial was considered successful if the encoder generated a collision-free trajectory that moved the end-effector between the specified poses while correctly manipulating the target object. The success rates for each motion encoder are reported in Table V.

TABLE V: Success rate of the motion encoders for skills produced for unseen start and goal states.

Skill	GCBC	DMP	GMM-GMR
PICK	9/15	15/15	14/15
PLACE	5/15	7/15	15/15
POUR	1/6	4/6	5/6

Discussion. As can be noted, the generalization capabilities differ from the reproduction one and expose the role of the structural priors embedded in each encoder. GMM-GMR is the most reliable across all three skills, thanks to its probabilistic encoding of the demonstrations, which produces plausible motions also when the start and goal poses leave the demonstrated distribution. DMP performs best on PICK (15/15) and remains competitive on POUR, since the critically damped attractor guarantees convergence to the new goal; its drop on PLACE (7/15) is consistent with the limits of the averaged-weight formulation, in which the forcing term does not reshape the trajectory when the start-goal displacement differs from the demonstrated one. GCBC is the weakest generalizer across all skills: lacking any explicit convergence mechanism, the per-step prediction errors of the goal-conditioned MLP accumulate over the rollout, and reliable extrapolation to unseen $(\mathbf{s}_0, \mathbf{g})$ pairs would require a substantially larger number of demonstrations than the one available in the proposed pipeline. This effect is most evident on the rotation-dominated POUR, where the orientation errors already observed in reproduction are amplified.

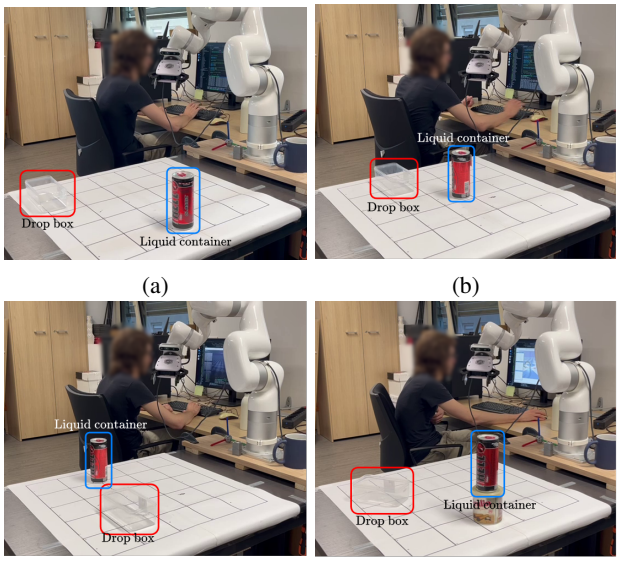
G. High-level tasks

The learned primitives are assembled into a high-level task effective to pour or drop something from a selected container into a receiving box, whose composition follows the programmatic scheme introduced in Section II-F. The corresponding pseudocode is reported in Algorithms 1, using the convention that each PRIMITIVE($\cdot$) call transparently triggers the query of the vision layer, the encoder-specific inference, and the execution on the manipulator. Gripper management is embedded in PICK (close) and PLACE (open).

Algorithm 1 POUR_DRINK

Require: $\text{ID}_{\text{drink}}, \text{ID}_{\text{glass}}, \texttt{quantity}$
1: PICK($\text{ID}_{\text{container}}$)
2: PLACE_TO_ALIGN($\text{ID}_{\text{drop\_box}}$)
3: POUR($\text{ID}_{\text{drop\_box}}$)

In order to test the generalization capabilities of each motion encoder, and the effectiveness of the learned skills when executed one after the other and not singularly, the high-level task in Algorithm 1 has been tested in four different scenarios. The starting configurations of the objects in such scenes are reported in Figure 5.



(a) (b) (c) (d)

Fig. 5: Starting configurations of the objects in the four different scenarios.

Table VI summarizes the success rate of the high-level task with the different motion encoders. Together with its TSR, the different SSR_σ are reported. It is worth noting that, due to the sequential nature of the composition, the failure of a low-level skill automatically propagates to all the subsequent ones and to the overall task, which are then counted as failed.

TABLE VI: Success rate of the motion encoders for the high-level skills.

	GCBC	DMP	GMM-GMR
TSR	1/4	0/4	3/4
SSR_{PICK}	3/4	4/4	3/4
SSR_{PLACE}	2/4	0/4	3/4
SSR_{POUR}	1/4	0/4	3/4

Discussion. The encoders used for composing high-level tasks confirm and amplify the trends already observed at the low-level skill stage. GMM-GMR is the only encoder able to reliably execute the full pipeline (3/4 TSR), with consistent per-skill success rates that reflect its accurate shape

reproduction and the robustness of its probabilistic encoding under unseen start–goal configurations. GCBC succeeds end-to-end only once (1/4 TSR): it manages most PICK executions but its endpoint imprecision, already observed in the low-level analysis, propagates downstream, since each primitive is initialized from the final state of the previous one and small terminal errors compromise the alignment required by PLACE and POUR. DMP exhibits the most striking drop with respect to the low-level results: despite a perfect $\mathrm{SSR}_{\texttt{PICK}}$ (4/4), the high-level task is never completed (0/4 TSR), because the averaged-weight formulation cannot reshape the trajectory when the start–goal displacement deviates from the demonstrated one, leading to systematic failures on PLACE that prevent any subsequent POUR. Overall, the high-level evaluation shows that endpoint precision alone is not sufficient for sequential composition, and that encoders combining accurate shape reproduction with robust generalization, such as GMM-GMR, are better suited to chaining low-level skills into multi-step tasks.

IV. Conclusion and future works

The paper presented a framework for learnable and reusable collaborative manipulation tasks, in which an LfD pipeline is implemented for low-level skill learning to execute higher-level tasks. Such a pipeline includes data collection to execution on a real robot.

Three complementary learning methods have been implemented and trained, in particular, GCBC for policy imitation, DMP for retargetable movement primitives, and GMM-GMR for probabilistic synthesis of the desired trajectory.

The overall system has been tested in a UFACTORY xArm6 cobot that combines an RGB-D sensor with ArUCO fiducial markers to parametrize skills to new dynamic goal poses while reducing the learning efforts.

The experimental results revealed that GMM-GMR is the most suitable encoder for skill composition, being the only one able to reliably complete the high-level task. The alternative methods, namely DMP and GCBC, despite their respective strengths in endpoint precision and shape reproduction, suffer more from error propagation across sequentially composed primitives. This confirms that, for chaining low-level skills into multi-step tasks, accurate shape reproduction combined with robust generalization is more relevant than endpoint precision alone.

Future works will include the exploration of an extension for task parametrization of the GMM and enrich the high-level learning with methods such as Behavior Trees, Finite State Machines, and Hidden Markov Models. Moreover, it could be interesting to explore learning low-level skills that involve periodic motions, so as to achieve a greater variety of real-world practical tasks.

References

[1] M. Tavassoli, S. Katyara, M. Pozzi, N. Deshpande, D. G. Caldwell, and D. Prattichizzo, "Learning skills from demonstrations: A trend from motion primitives to experience abstraction," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 1, pp. 57–74, 2023.

[2] A. D. Sosa-Ceron, H. G. Gonzalez-Hernandez, and J. A. Reyes-Avendaño, "Learning from demonstrations in human–robot collaborative scenarios: A survey," *Robotics*, vol. 11, no. 6, p. 126, 2022.

[3] A. Barekatain, H. Habibi, and H. Voos, "A practical roadmap to learning from demonstration for robotic manipulators in manufacturing," *Robotics*, vol. 13, no. 7, p. 100, 2024.

[4] W. Wang, C. Zeng, H. Zhan, and C. Yang, "A novel robust imitation learning framework for complex skills with limited demonstrations," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 3947–3959, 2024.

[5] Z. Wang, J. Chen, Z. Chen, P. Xie, R. Chen, and L. Yi, "Genh2r: Learning generalizable human-to-robot handover via scalable simulation demonstration and imitation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 362–16 372.

[6] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[7] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, and J. Kober, "Interactive imitation learning in robotics: A survey," *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.

[8] T. Tsuji, Y. Kato, G. Solak, H. Zhang, T. Petrič, F. Nori, and A. Ajoudani, "A survey on imitation learning for contact-rich tasks in robotics," *The International Journal of Robotics Research*, p. 02783649261417694, 2025.

[9] C. Li, X. Zhang, L. Chen, and J. Sun, "Learning from demonstrations—paradigms, challenges, and advances in human-robot skill transfer: A review," in *2025 International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2025, pp. 194–201.

[10] 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.

[11] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *Journal of machine learning research*, vol. 22, no. 30, pp. 1–82, 2021.

[12] 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, no. 1, pp. 297–330, 2020.

[13] A. Correia and L. A. Alexandre, "A survey of demonstration learning," *Robotics and Autonomous Systems*, vol. 182, p. 104812, 2024.

[14] S. Yang, W. Zhang, R. Song, J. Cheng, H. Wang, and Y. Li, "Watch and act: Learning robotic manipulation from visual demonstration," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 7, pp. 4404–4416, 2023.

[15] X. Liu, G. Wang, Z. Liu, Y. Liu, Z. Liu, and P. Huang, "Hierarchical reinforcement learning integrating with human knowledge for practical robot skill learning in complex multi-stage manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 3852–3862, 2023.

[16] C. Migliaccio, "Skill learning and task composition from human demonstrations for a collaborative manipulator," Master's thesis, Politecnico di Torino, 2025.

[17] S. Chernova and A. L. Thomaz, *Robot learning from human teachers*. Springer Nature, 2022.

[18] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *Advances in neural information processing systems*, vol. 32, 2019.

[19] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[20] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[21] UFACTORY, *UFACTORY xArm 6*, accessed: April 2026. [Online]. Available: https://www.ufactory.cc/xarm-collaborative-robot/#Working-Range

[22] "Github repository for experiments," [Accessed: April 2026]. [Online]. Available: https://github.com/celubi/HumRob-skill-task-composition.git

[23] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.