

Highlights

- DMP forcing term recast as a deep RL policy, refined from a reference trajectory.
- A spatial-scaling reformulation preserves classical-DMP generalisation.
- Multi-objective deep RL builds an imitation-versus-effort Pareto front.
- TOPSIS selects a balanced policy after a one-off objective normalisation.
- On hardware: 39% lower transport speed than its imitation-fidelity extreme.

Multi-objective Deep Reinforcement Learning of Dynamic Movement Primitives for Robotic Manipulation

Navid Noorgostar^{a,b,*}

^aDepartment of Mechanical Engineering, Faculty of Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran

^bDepartment of Computer Engineering, Faculty of Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran

Abstract

Learning from Demonstration (LfD) is an attractive way to transfer manipulation skills to industrial robots, but it is inherently imitative: the learnt skill is only as good as the trajectory it reproduces. Reinforcement learning (RL) can refine such a skill, yet suffers from high sample cost and the difficulty of hand-designing a single scalar reward. This paper presents a three-stage refinement framework coupling Dynamic Movement Primitives (DMPs) with deep RL, evaluated with prescribed analytic reference trajectories standing in for demonstrations; the pipeline is agnostic to the demonstration source. The DMP forcing term is recast as a policy network with a position/velocity-enriched state and a spatial-scaling reformulation that preserves classical-DMP generalisation. The policy is bootstrapped by supervised learning and refined with actor-critic deep RL (TD3, SAC, PPO). A prediction-guided multi-objective stage builds a Pareto front trading imitation fidelity against motion effort, and a TOPSIS step selects a balanced operating point without manual reward weighting, once the objectives share a common scale. The framework is validated on a pick-and-place task in MuJoCo and on a physical manipulator in three experiments totalling 850 trials. The selected policy lowers the integrated transport speed by 39.2% relative to the imitation-fidelity extreme of its Pareto front, at the highest task-success rate (250 trials); a condition-coverage variant transfers to unseen start, goal, payload, friction, height and obstacle variations with a 0.6-point success gap (360 trials); and the balanced policy cuts dynamic actuator effort, jerk and vibration by tens of percent at non-inferior accuracy (240 trials).

Keywords: Learning from demonstration, Dynamic movement primitives, Deep reinforcement learning, Multi-objective optimisation, Industrial robots, Sim-to-real validation

1. Introduction

The widespread industrial deployment of robotic manipulators is increasingly limited not by the mechanical capabilities of the hardware, but by the cost of programming the high-level skills that the robot is expected to perform. Conventional teach-pendant programming, while reliable for repetitive operations, scales poorly to flexible manufacturing scenarios in which the manipulator must execute a large number of variants of a task or quickly switch between tasks [1, 2]. Two complementary paradigms have therefore received considerable attention from the robotics community over the last decade: *Learning from Demonstration* (LfD), in which the robot acquires a skill by observing one or more human demonstrations [1, 3, 4], and *Reinforcement Learning* (RL), in which the robot improves a policy through trial-and-error interaction with the environment guided by a scalar reward [2, 5].

LfD methods are attractive because they require neither an analytical task model nor a hand-engineered reward function:

the user simply shows the robot what to do, e.g. through kinesthetic teaching, teleoperation or marker tracking, and the algorithm encodes the demonstration in a representation that can be reproduced on the robot [1, 6]. Among the many LfD representations proposed in the literature, *Dynamic Movement Primitives* (DMPs) [7, 8] have become a de-facto standard for trajectory-level encoding of point-to-point motions because of their attractive properties: convergence to a goal attractor, temporal and spatial scalability, robustness to perturbations and compactness of the underlying parameterisation. A central limitation of LfD methods, however, including DMP-based ones, is that they are inherently *imitative*: the learnt skill is, at best, as good as the demonstration. When the demonstration is sub-optimal or when the robot's morphology and dynamics differ from those of the demonstrator, the reproduced skill may be inefficient or even infeasible to execute [1, 6].

Reinforcement learning offers a natural way to refine an initial skill because the agent improves the policy by interacting with the environment and is therefore not bounded by the quality of the demonstration. Combined with deep function approximators, modern *deep RL* algorithms such as PPO [9], SAC [10] and TD3 [11] have produced impressive results on simulated continuous-control benchmarks. However, the direct application of deep RL to physical robotic manipulation is hampered by two well-known difficulties [2]. First, the high sample

*Corresponding author.

Email address: navid.noorgostar22@gmail.com (Navid Noorgostar)

complexity of deep RL makes from-scratch learning impractical when each interaction with the system is expensive or potentially damaging. Second, the agent’s performance is critically dependent on the design of a single scalar reward that must encode all relevant aspects of the task; in manipulation, this reward must simultaneously balance contradictory criteria such as task completion, accuracy with respect to a reference, smoothness, effort and safety, leading to brittle weight tuning.

A natural way to address both limitations is to combine LfD and RL: the demonstration is used to obtain an initial policy that solves the task well enough to bootstrap learning, and RL is then employed to improve this initial policy. While such combinations have been explored, most existing approaches still suffer from the second difficulty above because the reward function must be hand-engineered, and a successful weighting of competing criteria is found only through manual trial-and-error. Multi-objective RL [12] provides a principled way around this problem by treating the various criteria as separate objectives and returning a Pareto-optimal *set* of solutions, leaving the final operating point to the designer.

The present work proposes a complete *learning-control framework* for robotic manipulation that brings together these ideas under a single, modular mechatronic architecture, depicted in Fig. 1: a demonstration is encoded as a DMP, the DMP is exposed to a deep RL agent as a parameterised policy, the policy is refined against a vector of physically meaningful objectives by a multi-objective deep RL procedure, the resulting Pareto front is filtered by a TOPSIS multi-criteria decision-making step, and the selected primitive is executed on the manipulator through an inverse-kinematics layer and a joint-space controller closed around the physical or simulated robot. The integration is therefore vertical, from the demonstration interface down to the joint torques, rather than algorithmic in isolation. The framework is validated on canonical sine and cosine reference trajectories, on a pick-and-place case study performed in the MuJoCo simulator with a collaborative manipulator, and, to close the loop with real hardware, on a physical manipulator in three experiments: a 250-trial pick-and-place study, a 360-trial cross-condition robustness study and a 240-trial actuator-effort study. Throughout, the demonstrations are prescribed analytic reference trajectories rather than human-recorded motions; the pipeline is agnostic to the source of the demonstration (Section 5.1), so “demonstration” is used here in the sense of a reference trajectory to be reproduced and refined, and the framing is one of trajectory refinement rather than a claim about a physical teaching interface. The contributions of the paper can be summarised as follows:

1. **Integrated learning-control architecture.** A modular six-block architecture (demonstration → DMP → deep policy → MORL/TOPSIS → inverse kinematics → joint controller → robot) is proposed that exposes a clean interface between the trajectory-generation layer (where learning happens) and the motion-execution layer (where the mechatronic constraints are enforced).
2. **DMP as a deep policy, with an analysed state design.**

The DMP forcing term is reformulated as a deep parameterised policy over the canonical phase, enriched with the current position and velocity of the tracked trajectory so that the imitation reward becomes a function of the observed state and so that the spatial-scaling reformulation of the next contribution has the inputs it operates on. We deliberately do *not* present this enrichment as an accuracy device: the ablation of Section 5.8 shows that in deterministic single-trajectory imitation the phase-only policy is sufficient and the enrichment can even cost accuracy through a covariate shift that we identify and explain. The enriched state earns its place where rollouts depart from the reference — stochastic exploration, perturbations, and changed start-goal pairs (Sections 4.1.3 and 5.10).

3. **Spatial-scaling reformulation.** A position-scaling reformulation is introduced that preserves the spatial-generalisation property of classical DMPs when the start and goal of the motion differ from those used during training, despite the fixed weights of the policy network.
4. **Supervised bootstrap of the RL stage.** A supervised pre-training procedure that exploits the analytical inverse of the DMP equation initialises the policy from the demonstration so that subsequent deep RL refinement starts from a feasible solution rather than from a random one.
5. **Multi-objective refinement with restructured tuning.** The improved primitive is obtained through a prediction-guided multi-objective deep RL procedure that constructs a discrete Pareto front of the imitation-vs-effort trade-off, and a TOPSIS-based selection step then picks a balanced operating point *after* the front has been built. Rather than eliminating tuning outright, this restructures it: the brittle search over scalar reward weights is replaced by a one-off objective-normalisation step (Section 4.3) and a single interpretable preference vector applied a posteriori.
6. **Algorithmic and system-level validation.** Three popular deep RL back-ends (TD3, SAC, PPO) are compared on canonical trajectories with a clear sample-efficiency-vs-wall-clock trade-off, the framework is extended in closed form to full six-degree-of-freedom task-space motions, and the complete pipeline is demonstrated on a pick-and-place case study in MuJoCo with a PD-controlled inverse-kinematics interface and subsequently validated on a physical manipulator in three hardware experiments totalling 850 evaluation trials (plus a matched 240-trial reference-condition baseline in Experiment 2, giving 1090 physical trials in all): a 250-trial, five-controller study in which the automatically selected balanced policy reproduces the simulated velocity-reduction benefit (39.2%) while achieving the highest task-success rate, a 360-trial cross-condition study showing that augmenting the multi-objective refinement with condition coverage transfers the skill robustly across unseen operating conditions (a 0.6-point success transfer gap), and a 240-trial effort study showing that the balanced policy reduces the dynamic actuator effort, jerk

and vibration by tens of percent at non-inferior accuracy, providing a reproducible end-to-end mechatronic realisation that spans simulation and real hardware.

The remainder of the paper is organised as follows. Section 2 surveys related work in LfD, DMPs and the combination of LfD with RL. Section 3 reviews the DMP formalism and the deep RL algorithms used in the experiments. Section 4 presents the proposed framework, including the integrated system architecture, the policy parameterisation, the supervised pre-training step, the multi-objective refinement, the TOPSIS-based selection, the six-degree-of-freedom extension and consolidated algorithmic pseudocode. Section 5 describes the implementation, presents results on canonical trajectories, compares the three deep RL back-ends, reports the pick-and-place case study in MuJoCo and its validation on a physical manipulator (including cross-condition robustness and actuator-effort studies), and includes ablation, sensitivity, generalisation and obstacle-avoidance studies. Section 6 concludes the paper and discusses directions for future work.

2. Related Work

This section reviews the two literatures that underlie the proposed framework: learning from demonstration in robotics, with particular emphasis on movement-primitive representations, and the various attempts that have been made to combine demonstrations with reinforcement learning so as to improve the quality of the learnt skill.

2.1. Learning from Demonstration in Robotics

Learning from Demonstration, also referred to in the literature as imitation learning, programming by demonstration or apprenticeship learning, encompasses a wide range of methods that allow a robot to acquire a skill from a set of demonstrations rather than from explicit programming [1, 3]. Three main demonstration modalities are commonly distinguished: *kinaesthetic teaching*, in which the human physically guides the robot’s end-effector [4, 13]; *teleoperation*, in which the demonstration is performed on the same kinematic structure as the executing robot, frequently with the help of virtual-reality interfaces [14, 15, 16, 17]; and *observation*, in which the robot watches a human performing the task using vision or motion capture [3, 18, 19]. The first two modalities largely avoid the so-called *correspondence problem* between human and robot bodies [20], while observation is more natural for the human teacher but requires the additional step of mapping the demonstration from the teacher’s body to the robot’s.

LfD methods can be further classified by the level at which the demonstration is encoded. *Planning-based* methods extract task constraints or cost functions from the demonstration and use these to plan new trajectories, e.g. through sequential convex optimisation [21, 22], stochastic trajectory optimisation [23, 24], or general movement-primitive optimisation [25]. *Inverse reinforcement learning* methods estimate a reward function that explains the demonstration and then solve the underlying RL problem [26, 27, 28, 29]. *Direct policy or trajectory encoding* methods, in contrast, encode the demonstrated

movement itself in a parameterised representation that can be reproduced and, to some extent, generalised. Gaussian Mixture Models with Gaussian Mixture Regression [30], Probabilistic Movement Primitives [31] and Dynamic Movement Primitives (DMPs) [7, 8, 6] fall into this last category.

DMPs, introduced by Ijspeert et al. [7, 8], encode a movement as a stable second-order dynamical system whose attractor is a user-supplied goal and whose transient is shaped by a learnt forcing term. These properties (Section 1) have made them the most widely used trajectory-level representation in modern LfD [1, 6]. They are also used industrially: Ti et al. [32], for example, generalise both orientation trajectories and force–torque profiles within the DMP framework to transfer a human peg-in-hole assembly skill to a robot under variable task constraints. Several authors have proposed variants that overcome specific drawbacks of the original formulation: Park et al. [33] added an extra term that removes the singularity occurring when the start and goal coincide; Hoffmann et al. [34] introduced biologically inspired DMP variants with online obstacle avoidance; and Ginesi et al. [35] consolidated these improvements in the so-called DMP++ formulation. Among the alternative LfD encodings, Pervez and Lee [36] proposed task-parameterised DMPs based on a mixture of Gaussians, and Maeda et al. [31] developed ProMPs for human–robot collaborative tasks. A long-standing limitation of all these methods, however, is that the learnt skill is essentially a reproduction of the demonstration; without additional mechanisms there is no way to refine the skill if the demonstration is suboptimal or if the robot’s dynamics differ from those of the demonstrator [1, 6].

2.2. Combining Demonstration with Reinforcement Learning

To overcome the imitation barrier of LfD, several authors have explored combinations of demonstrations with RL in which the demonstration serves as an initial policy that is then improved by interaction with the environment [2]. Peters and Schaal [37] first showed that DMPs can be improved using natural-actor–critic policy-gradient methods. The Policy Improvement with Path Integrals algorithm of Theodorou et al. [38] provides a probabilistic gradient-free alternative that has been widely adopted in DMP refinement [39]. Stulp and Schaal [39] extended this approach to a hierarchical setting in which an upper-level policy selects DMP shape parameters and a lower-level policy executes the resulting motion. More recently, Kim et al. [40] proposed a hierarchical deep RL architecture that decomposes a complex task into a sequence of DMPs and learns both the sequencing and the shaping of the primitives from demonstration, while Kim et al. [41] showed that movement-primitive-based RL is particularly suited to contact-rich assembly tasks. Hierarchical RL ideas have also been used in pure-RL settings without demonstrations [42, 43, 44, 45]. Mohseni-Kabir et al. [46] learn primitives and their hierarchy simultaneously, and several authors have studied the segmentation and sequencing of primitives [47, 48, 49, 50, 51, 52, 53, 54].

Despite the maturity of these approaches, a common practical difficulty has remained: when the refinement objective itself involves several competing criteria — for instance, the

trade-off between staying close to the demonstration and reducing motion effort, jerk or execution time — the user must combine these criteria into a single scalar reward by manually choosing weights. The optimal weighting depends on the specific demonstration and on the robot, and is typically found through trial-and-error, which limits the practical scalability of LfD-plus-RL pipelines. *Multi-objective* RL [12] replaces the single scalar reward with a vector reward and returns a Pareto-optimal set of policies; the final operating point is then chosen *a posteriori* by the designer rather than *a priori* through weight tuning. Xu et al. [12] recently proposed a prediction-guided algorithm that constructs a high-quality Pareto front for continuous-control problems within a deep RL framework. Recent work also combines the two ingredients at the level of task structure rather than of a single motion: Rauso et al. [55] pair imitation learning from virtual demonstrations with reinforcement learning to train reusable manipulation primitives incrementally, then compose them symbolically into structured tasks. That line is complementary to the present one, which refines a single continuous primitive against competing continuous objectives rather than sequencing discrete ones. Closest to the construction used here, Li et al. [56] embed a DMP-based policy in the actor-critic structure of DDPG and derive the corresponding update rules, so that the networks learn to set the *parameters* of the DMP. The present work differs in two respects: the policy replaces the forcing term itself and is queried at every time step rather than emitting a parameter vector once per trajectory, and the refinement is multi-objective, so that the operating point is chosen *a posteriori* from a Pareto front instead of being fixed in advance by a single scalar reward. The present work combines this multi-objective deep RL approach with DMP-based LfD and adds a TOPSIS [57] multi-criteria decision-making step that picks a balanced policy from the front automatically, thereby producing a fully automated improvement pipeline that does not require the user to weight the objectives by hand.

3. Background

For completeness, this section briefly summarises the two building blocks of the proposed framework: the Dynamic Movement Primitive formulation used to represent the demonstrated motion, and the deep reinforcement learning algorithms used as the back-end of both the imitation and the refinement stages. The notation introduced here is used throughout the rest of the paper.

3.1. Dynamic Movement Primitives

A Dynamic Movement Primitive encodes a one-dimensional motion from an arbitrary start point to a fixed goal g as a stable, non-linear, second-order dynamical system. The model consists of two coupled differential equations, often referred to as the *transformation* system, together with a third differential equation that drives the system through time, called the *canonical* system. Following the formulation of Park et al. [33], which is the variant used in this work because it removes the

start-goal coincidence singularity of the original DMP, the transformation system is

$$\tau \dot{z} = K(g - y) - Dz - K(g - y_0)x + Kf(x), \quad (1)$$

$$\tau \dot{y} = z, \quad (2)$$

where y is the position of the modelled coordinate, z is a scaled velocity, τ is a temporal scaling constant, K and D are stiffness and damping gains satisfying the critical-damping condition $D = 2\sqrt{K}$, g is the goal position, y_0 is the start position, and $f(x)$ is a non-linear *forcing term* that shapes the transient of the motion. The variable x is the phase of the motion, generated by the canonical system

$$\tau \dot{x} = -\alpha_x x, \quad x(0) = 1, \quad (3)$$

where $\alpha_x > 0$ is chosen so that x decays from 1 to a value close to 0 over the duration of the motion. Replacing time by the phase x removes the explicit time dependence of the transformation system, which is the key property that allows a DMP to be temporally rescaled by changing τ without retraining.

The forcing term is parameterised as a normalised weighted sum of N radial basis functions of the phase variable,

$$f(x) = \frac{\sum_{i=1}^N \psi_i(x) \omega_i}{\sum_{i=1}^N \psi_i(x)} x, \quad \psi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\sigma_i^2}\right), \quad (4)$$

where c_i and σ_i are the centre and width of the i -th basis function and ω_i are the trainable weights. The multiplicative phase factor x in (4) guarantees that $f(x) \rightarrow 0$ as the motion completes, so that the transformation system converges to the goal g as a stable second-order linear system.

Given a demonstrated trajectory $\{y_{\text{demo}}(t), \dot{y}_{\text{demo}}(t), \ddot{y}_{\text{demo}}(t)\}_{t=1}^P$ sampled at P time steps, learning a DMP amounts to estimating the weights ω_i so that the forcing term explains the residual between the demonstration and the linear goal-attractor system. The target forcing function is obtained from the inverse of (1)–(2),

$$f_{\text{target}}(t) = \frac{1}{\tau^2} \left[\tau^2 \ddot{y}_{\text{demo}}(t) + D\tau \dot{y}_{\text{demo}}(t) - K(g - y_{\text{demo}}(t)) + K(g - y_0)x(t) \right], \quad (5)$$

and the weights are obtained in closed form from a locally weighted regression problem,

$$J_i = \sum_{t=1}^P \psi_i(t) (f_{\text{target}}(t) - \omega_i \xi(t))^2, \quad \xi(t) = x(t), \quad (6)$$

whose minimiser is

$$\omega_i = \frac{\mathbf{s}^T \mathbf{\Gamma}_i \mathbf{f}_{\text{target}}}{\mathbf{s}^T \mathbf{\Gamma}_i \mathbf{s}}, \quad (7)$$

with $\mathbf{s} = [\xi(1), \dots, \xi(P)]^T$, $\mathbf{\Gamma}_i = \text{diag}(\psi_i(1), \dots, \psi_i(P))$ and $\mathbf{f}_{\text{target}} = [f_{\text{target}}(1), \dots, f_{\text{target}}(P)]^T$. Consistently with the Park transformation system (1), the forcing is regressed on the phase alone (and is therefore *not* amplitude-scaled by $g - y_0$), while the start-goal offset is carried by the explicit $-K(g - y_0)x$ term; the same convention underlies the learning target (5) and the supervised target (19) used later.

The closed-form solution (7) is sometimes referred to as *batch learning* because it requires the entire demonstration to be available. The classical DMP framework is therefore inherently imitative: once (7) has been solved, the only way to improve the policy is to either provide a better demonstration or to combine the DMP with an additional refinement loop such as the RL-based loop proposed in this paper.

The same construction applies independently to each component of a D -dimensional task-space trajectory, so that the multi-dimensional DMP is simply a vector of identical scalar DMPs sharing the same canonical system [8].

3.2. Deep Reinforcement Learning

Reinforcement learning casts sequential decision-making as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p(s_{t+1} | s_t, a_t)$ is the transition kernel, $r(s_t, a_t)$ is a scalar reward and $\gamma \in [0, 1)$ is a discount factor [5]. A policy $\pi_\theta(a | s)$ parameterised by θ is sought that maximises the expected discounted return $J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_t \gamma^t r(s_t, a_t)]$. In continuous-action robotic settings, *actor-critic* algorithms combine a parametric policy (actor) with a parametric value or action-value function (critic) to obtain low-variance policy-gradient estimates. Three actor-critic algorithms are used in this work, all of which are widely available in standard deep-RL libraries and are briefly recalled below.

Proximal Policy Optimization (PPO). PPO [9] is an on-policy stochastic-policy-gradient method that maximises a clipped surrogate objective so that each update remains within a trust region around the previous policy. The clipping mechanism makes PPO empirically more stable than vanilla policy gradient and trust-region methods such as TRPO [58], while remaining straightforward to implement.

Soft Actor-Critic (SAC). SAC [10] is an off-policy maximum-entropy actor-critic algorithm. By augmenting the standard return with an entropy bonus, SAC encourages exploration and frequently achieves higher sample efficiency than purely deterministic methods on benchmark control tasks. The use of a replay buffer and two Q-function estimators (to address overestimation bias) make SAC robust but computationally heavier per environment step than on-policy methods.

Twin Delayed DDPG (TD3). TD3 [11] is an off-policy deterministic-policy actor-critic algorithm that improves upon Deep Deterministic Policy Gradient (DDPG) [59] by introducing three mechanisms that mitigate the well-known overestimation bias of actor-critic methods: a pair of critic networks whose minimum is used as the target, delayed policy updates relative to the critic, and target-policy smoothing through noise injection.

In the proposed framework, any of these three algorithms can be used to optimise the DMP-as-policy of Section 4. Section 5 compares them empirically and reports the trade-off between sample efficiency (number of episodes to convergence) and wall-clock cost (training time on the same hardware).

3.3. Multi-objective Reinforcement Learning

In multi-objective RL, the scalar reward r is replaced by a vector reward $\mathbf{r} \in \mathbb{R}^m$, one component per objective, and the agent’s value function becomes a vector $\mathbf{V}^\pi(s) \in \mathbb{R}^m$ [12]. Given a non-negative weight vector $\omega \in \mathbb{R}_{\geq 0}^m$ with $\sum_i \omega_i = 1$, the corresponding scalar problem is

$$\max_{\theta} J(\theta, \omega) = \omega^\top \mathbf{F}(\pi_\theta) = \sum_{i=1}^m \omega_i J_i(\pi_\theta), \quad (8)$$

with $J_i(\pi_\theta)$ the expected return for the i -th objective. Sweeping ω across the simplex traces an approximation of the *Pareto front* of the multi-objective problem, i.e. the set of policies for which no objective can be improved without degrading at least one other.

Two scalar indicators are widely used to assess the quality of a discrete Pareto-front approximation [12]: the *hypervolume* H that measures the size of the dominated region with respect to a reference point and reflects both convergence and spread; and the *spread* (sparsity) indicator S of Xu et al. [12], defined as the average, over the m objectives, of the mean squared gap between consecutive front points sorted along that objective. Note the sense of S : unlike H , *smaller* values are better, a small S indicating a front whose points are densely and evenly distributed rather than clustered with large gaps. Throughout this paper the hypervolume reference point is taken as the component-wise nadir of the evaluated objectives (the worst value attained on each axis), and the objectives are placed on a fixed common scale before H and S are computed. Because both indicators depend on this reference point and on the objective scaling — S quadratically, as it accumulates squared gaps — H and S values are comparable only across fronts computed on the same scale; the scale used for each reported value is stated where it appears. These two indicators are reported in Section 5 to characterise the Pareto fronts produced by the proposed framework.

4. Proposed Framework

The proposed framework integrates Dynamic Movement Primitives with deep reinforcement learning in three sequential stages, as depicted in Fig. 2, and is implemented as the modular learning-control architecture shown in Fig. 1. *Stage I* (Section 4.1) reformulates the DMP as a parameterised policy that is trained, by supervised learning followed by deep RL, to imitate the demonstration. *Stage II* (Section 4.2) embeds the trained primitive into a prediction-guided multi-objective deep RL procedure that returns a discrete approximation of the Pareto front for a vector of competing objectives, typically imitation fidelity versus motion effort. *Stage III* (Section 4.3) selects a single operating point from the Pareto front through TOPSIS, a well-established multi-criteria decision-making algorithm. Section 4.4 discusses the extension of the framework to full six-degree-of-freedom task-space motions, and Section 4.5 consolidates the entire pipeline into algorithmic pseudocode. The overall pipeline (Section 4.7) requires no manual reward-weight tuning — the weighting is replaced by a documented

objective-normalisation step and a single interpretable TOPSIS preference vector (Section 4.3) — and produces a refined, executable DMP from a single demonstration.

4.1. Stage I — Imitative Learning of the DMP

4.1.1. DMP as a parameterised policy

Because the canonical system (3) is deterministic and the transformation system (1)–(2) is fully specified once the forcing term f is known, learning a DMP can be cast as a reinforcement-learning problem in which the forcing term *is* the policy. To make this construction concrete, we discretise the canonical and transformation systems with an Euler scheme of step size T and treat each step as one transition of an MDP. The canonical update becomes

$$x_{t+1} = x_t - T \alpha_x x_t / \tau, \quad x_0 = 1, \quad (9)$$

and the trajectory update is

$$\begin{aligned} \ddot{y}_t &= \frac{1}{\tau^2} [K(g - y_t) - D\tau\dot{y}_t - K(g - y_0)x_t + Kf_t], \\ \dot{y}_{t+1} &= \dot{y}_t + T \ddot{y}_t, \\ y_{t+1} &= y_t + T \dot{y}_t, \end{aligned} \quad (10)$$

in which f_t is the policy output at the current step.

A naive choice would be to parameterise f_t directly as the output of a neural network h_θ , i.e. $f_t = h_\theta(x_t)$. Two issues, however, must be addressed before this construction can be deployed.

Convergence to the goal. The DMP relies on $f(x) \rightarrow 0$ as $x \rightarrow 0$ so that the transformation system converges to the goal as a critically damped second-order linear system. If $h_\theta(x_t)$ is the output of a neural network with non-zero bias, this convergence is no longer guaranteed and the end-effector can drift away from the goal at the end of the motion. We therefore parameterise the policy through the auxiliary function h_θ *multiplied by the phase*, so that the action emitted by the agent is

$$f_t = h_\theta(\cdot) x_t, \quad (11)$$

where the argument of h_θ is to be defined below. Since $x_t \rightarrow 0$ at the end of the motion, this guarantees $f_t \rightarrow 0$ and preserves the goal-convergence property of the DMP without any additional constraint on the network weights.

4.1.2. State-space enrichment for the Markov property

If the only state variable were the phase x_t , the imitation reward would not in general be a function of the observed state: given x_t and f_t , the next position y_{t+1} cannot be predicted without also knowing y_t and $\dot{y}_t$, which are not recoverable from x_t across policies. This does not formally violate the deterministic-MDP assumption — for a fixed deterministic policy and initial condition the rollout, and hence the reward, is indexed by the phase alone — but it does mean that the value function cannot distinguish rollouts that share a phase yet differ in position, which matters as soon as the rollout distribution departs from the reference through stochastic exploration, perturbations, or a changed start–goal pair. We therefore enrich

the state to include the position and velocity of the tracked trajectory,

$$s_t = (x_t, y_t, \dot{y}_t), \quad (12)$$

and let the policy network receive the full state as input. The action is still (11) but the auxiliary function now depends on the enriched state,

$$f_t = h_\theta(x_t, y_t, \dot{y}_t) x_t. \quad (13)$$

The resulting MDP has continuous state and action spaces and a deterministic transition kernel given by (9)–(10); it is therefore directly compatible with the actor–critic algorithms recalled in Section 3.2. The flip side of this choice is quantified in Section 5.8: in strictly deterministic single-trajectory imitation the enrichment is unnecessary and can cost accuracy through co-variate shift, so it should be read as the structural interface required by the spatial-scaling reformulation of Section 4.1.3, not as an accuracy device.

4.1.3. Spatial-scaling reformulation

A key property of the classical DMP is its spatial scalability: once the weights ω_i have been learnt for a particular start–goal pair (y_0, g) , the same weights produce a shape-preserving trajectory for any other pair (y'_0, g') , with the trajectory being affinely scaled by the ratio of the two amplitudes. With a fixed-weight neural-network policy, this property is no longer automatic, because the network outputs h_θ depend on the absolute values of the inputs y_t and $\dot{y}_t$ rather than on their position relative to the start and the goal. To recover the spatial-scalability property we propose an input/output normalisation scheme that turns the deep policy into a *shape-invariant* controller.

Let the original demonstration have start y_0 and goal g , and let (y'_0, g') denote a new start–goal pair to which the learnt motion must be transferred. Following the spatial-scaling argument of Ijspeert et al. [8], the desired transformed trajectory should satisfy

$$\frac{y'(t) - y'_0}{y(t) - y_0} = \frac{g' - y'_0}{g - y_0} \triangleq A, \quad (14)$$

so that the scaled position and velocity become

$$y'(t) = A(y(t) - y_0) + y'_0, \quad \dot{y}'(t) = A\dot{y}(t). \quad (15)$$

Substituting (15) into the DMP relation (1) and rearranging yields the scaled transformation system

$$\begin{aligned} \tau \dot{z}' &= K(g' - y') - Dz' - K(g' - y'_0)x \\ &\quad + KA h_\theta(x, \mathcal{N}_p^{-1}(y'), \mathcal{N}_v^{-1}(\dot{y}'))x, \end{aligned} \quad (16)$$

which shows that the same policy h_θ can be re-used at run time provided its position and velocity inputs are first mapped back to the training frame by the inverse normalisers $\mathcal{N}_p^{-1}$ and $\mathcal{N}_v^{-1}$ and its forcing output is post-scaled by $\mathcal{N}_f$. With the amplitude ratio A of (14), the forward maps (training frame $\rightarrow$ deployment frame) are

$$\mathcal{N}_p(y) = A(y - y_0) + y'_0, \quad \mathcal{N}_v(\dot{y}) = A\dot{y}, \quad \mathcal{N}_f(f) = Af, \quad (17)$$

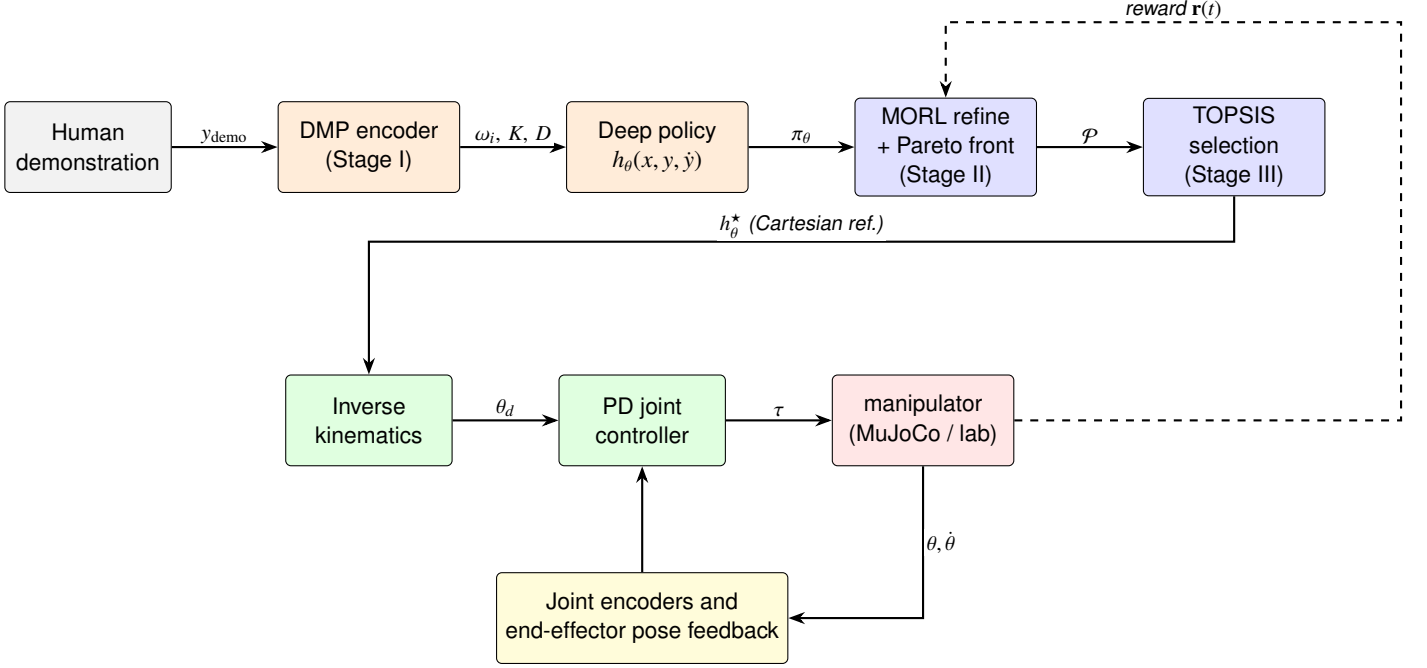


Figure 1: Integrated learning-control architecture of the proposed framework. The upper row implements the trajectory-generation (learning) layer: a demonstration is encoded by the DMP, its forcing term is parameterised as a deep policy, the policy is refined by multi-objective deep RL into a Pareto front of candidate primitives, and TOPSIS selects a single operating point h_θ^* . The lower row is the motion-execution (mechatronic) layer: the selected Cartesian reference is mapped to joint references θ_d by an inverse-kinematics solver, tracked by a PD joint controller that emits the joint torque τ and applied to the manipulator (simulated or physical), with joint encoders closing the inner motion loop. The dashed arrow returns the simulated/measured state to the multi-objective stage as the vector reward $\mathbf{r}(t)$, closing the outer learning loop.

and the inverse maps (deployment frame $\rightarrow$ training frame) used in (16) are

$$\mathcal{N}_p^{-1}(y') = A^{-1}(y' - y'_0) + y_0, \quad \mathcal{N}_v^{-1}(\dot{y}') = A^{-1}\dot{y}'. \quad (18)$$

Equations (17)–(18) thus implement the spatial-scalability property at the controller level without modifying the policy weights, and are applied as a pre- and post-processing stage whenever the start-goal pair differs from the one used during training. The normalisers $\mathcal{N}_\bullet$ are distinct from the stiffness gain K of (1) and from the PD gains of the joint controller.

4.1.4. Supervised pre-training of the policy

Although the MDP formulation of Section 4.1.1–4.1.2 is now fully compatible with standard actor-critic algorithms, training the policy from scratch is wasteful because the demonstration itself contains all the information required to obtain a near-optimal initialisation. By inverting the discretised transformation system (10) with respect to h_θ , the per-step supervision target is

$$h_{\text{target}}(t) = \frac{\tau^2 y_{\text{demo}}''(t) + D\tau y_{\text{demo}}'(t) - K(g - y_{\text{demo}}(t)) + K(g - y_0)x_t}{Kx_t}. \quad (19)$$

A supervised regression problem with input-output pairs

$$((x_t, y_{\text{demo}}(t), y_{\text{demo}}'(t)), h_{\text{target}}(t))_{t=1}^P \quad (20)$$

is then solved with the same network architecture that will be used in the subsequent RL phase, by mini-batch stochastic gradient descent. The pre-trained network is used to initialise the

actor of the RL algorithm; the critic is initialised randomly and refined together with the actor during the RL phase.

4.1.5. Reward and deep RL refinement

The reward used in the imitation phase is the negative absolute deviation between the policy-generated position and the demonstration at each time step,

$$r(t) = -|y_t - y_{\text{demo}}(t)|, \quad (21)$$

which yields a return whose maximum is zero when the policy reproduces the demonstration exactly. Any of the three actor-critic algorithms reviewed in Section 3.2 can be used to optimise the pre-trained policy under this reward; Section 5 reports a head-to-head comparison and motivates the choice of PPO as the back-end of the case study. At the end of Stage I, a deterministic policy $h_{\theta_{\text{init}}}^*$ that reproduces the demonstrated trajectory to within the residual error inherent to the chosen number of basis functions and network capacity is obtained.

4.2. Stage II — Multi-objective Refinement

The objective of Stage II is to refine the imitative primitive against a vector of m objectives that captures, in addition to imitation, complementary criteria such as motion effort, jerk, execution time, or distance to obstacles. The vector reward used in this work has two components,

$$\mathbf{r}(t) = \begin{bmatrix} -|y_t - y_{\text{demo}}(t)| \\ -|\dot{y}_t| \end{bmatrix}, \quad (22)$$

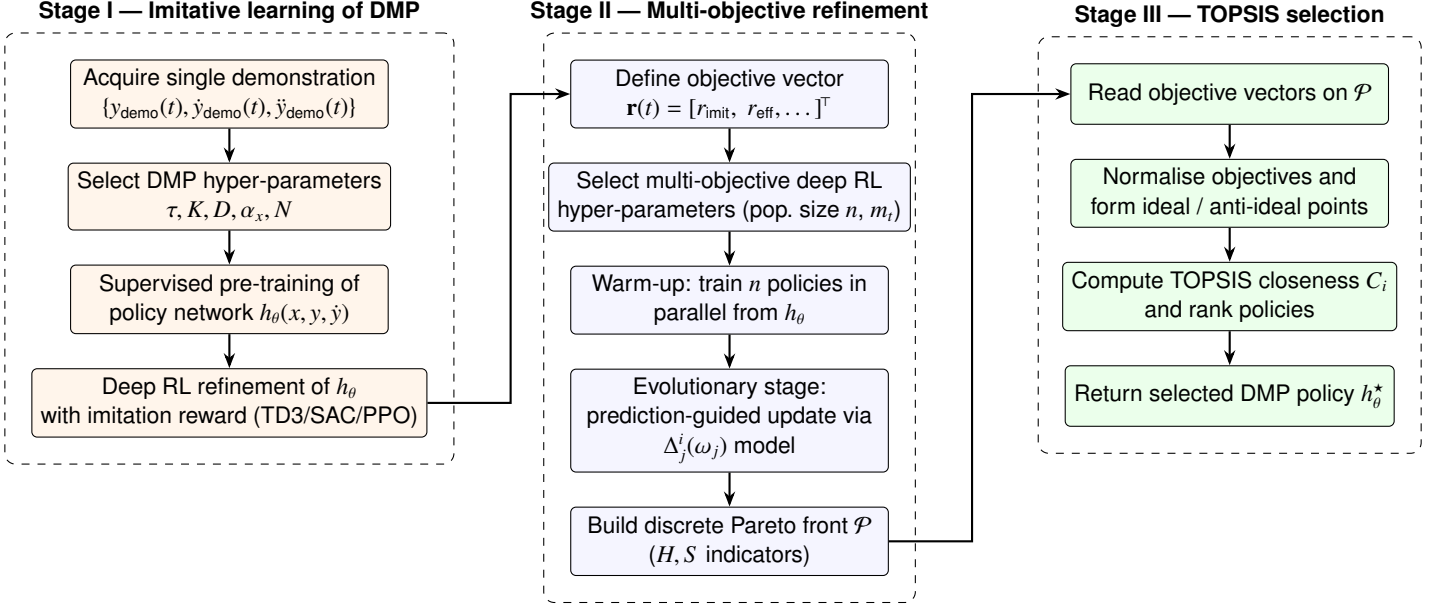


Figure 2: Block diagram of the proposed three-stage framework. Stage I (left) learns an imitative DMP policy from a single demonstration via supervised pre-training followed by deep RL. Stage II (centre) refines the primitive against a vector of objectives using prediction-guided multi-objective deep RL and constructs a discrete Pareto front. Stage III (right) selects a balanced operating point on the front using TOPSIS multi-criteria decision making.

where the first component drives the policy towards the demonstration (imitation) and the second component penalises high end-effector velocities (motion effort or, equivalently, execution speed). The same construction generalises immediately to additional or alternative objectives.

4.2.1. Multi-objective policy gradient

For a non-negative weight vector ω on the unit simplex, the scalarised objective is given by (8) and the corresponding policy gradient becomes [12]

$$\nabla_{\theta} J(\theta, \omega) = \mathbb{E} \left[\sum_{t=0}^{T_e} \omega^{\top} \mathbf{A}^{\pi}(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right], \quad (23)$$

in which $\mathbf{A}^{\pi}(s_t, a_t) \in \mathbb{R}^m$ is a vector of advantage functions, one per objective, and T_e is the episode length. Because the weight vector ω is allowed to change between updates, a single scalar value function cannot be reused; instead, a *vector-valued* value function $\mathbf{V}^{\pi}(s) \in \mathbb{R}^m$ is maintained and trained by minimising $\|\mathbf{V}^{\pi}(s) - \hat{\mathbf{V}}(s)\|^2$ where $\hat{\mathbf{V}}(s)$ is the multi-objective bootstrap target. Equation (23) is written for a *stochastic* policy and therefore applies directly to the PPO and SAC back-ends. For the deterministic TD3 back-end (whose Stage I output is a deterministic policy), the corresponding *deterministic* policy gradient [59, 11] is used instead, in which the score-function term $\nabla_{\theta} \log \pi_{\theta}$ is replaced by back-propagation through a vector-valued action-value function, $\nabla_{\theta} J(\theta, \omega) = \mathbb{E} [\omega^{\top} \nabla_a \mathbf{Q}^{\pi}(s, a)|_{a=\mu_{\theta}(s)} \nabla_{\theta} \mu_{\theta}(s)]$, with μ_{θ} the deterministic actor and $\mathbf{Q}^{\pi} \in \mathbb{R}^m$ the per-objective critics; the back-end-agnostic claim of the framework is to be read in this sense.

4.2.2. Prediction-guided population update

A naive sweep of ω across the simplex would require an unaffordable number of independent policy-gradient runs. The prediction-guided algorithm of Xu et al. [12], adopted in this work, avoids this cost by building, alongside each policy in the current population, a small analytical model that predicts the per-objective improvement attainable after m_t gradient updates as a function of the chosen weight ω_j . For a policy π_i improving objective f_j with weight ω_j , the prediction model is

$$\Delta_j^i(\omega_j) = A \frac{e^{a(\omega_j-b)} - 1}{e^{a(\omega_j-b)} + 1} + c, \quad (24)$$

whose four parameters A, a, b, c are estimated, for each (i, j) pair, from a small buffer $\mathcal{R}$ of past triples $(\mathbf{F}(\pi), \mathbf{F}(\pi'), \omega)$ stored from previous iterations using non-linear least squares. The intuition behind (24) is that a small weight on an objective produces a small improvement, a large weight produces a saturating improvement, and the transition between the two regimes is smooth.

Given the prediction model and the current discrete Pareto front EP, the algorithm selects the n next pairs $\mathcal{T} = \{(\pi_i, \omega_i)\}_{i=1}^n$ that maximise a quality functional

$$\max_{\mathcal{T}} Q(\text{EP}, \mathcal{T}) = \mathcal{H}(P) + \alpha S(P), \quad (25)$$

with $P = \text{Pareto}(\mathbf{F}(\text{EP}) \cup \{\mathbf{F}(\pi_i) + \Delta^i(\omega_i)\})$,

where $\mathcal{H}(\cdot)$ and $S(\cdot)$ are the hypervolume and sparsity of the predicted Pareto front and $\alpha \geq 0$ trades the two indicators. Equation (25) is a mixed-integer programme that is solved efficiently by discretising the simplex into K samples and reducing the resulting selection to a knapsack problem [12]. Each selected pair is then improved by m_t multi-objective

policy-gradient updates using (23), and the resulting policies are merged into the next population.

4.2.3. Continuous Pareto-front representation

After a fixed number of generations of the evolutionary step, the obtained set of Pareto-optimal policies is parameterised so that an operating point can be selected for any vector of preferences. The dimensionality of the policy parameters is first reduced by t-SNE [60], and the resulting low-dimensional embedding is partitioned by k -means [61]. For an arbitrary objective vector, the cluster to which it belongs is identified and the corresponding policy parameters are interpolated within that cluster, producing a continuous Pareto-front representation. This continuous representation is an *optional* convenience for querying preferences that fall between the discretely sampled policies; it is not required by Stages I–III and is not exercised in the experiments of Section 5, which select operating points directly from the discrete front. We note, moreover, that the t-SNE embedding is non-parametric and does not preserve global distances or admit out-of-sample points, so a parametric, directly invertible embedding would be preferable in any deployment that relies on this step; it is included here as a preliminary component rather than a validated contribution.

4.3. Stage III — Operating-Point Selection via TOPSIS

Once the discrete Pareto front $\mathcal{P} = \{\mathbf{F}_i\}_{i=1}^{N_p} \subset \mathbb{R}^m$ has been built, the user is still left with the multi-criteria decision-making problem of choosing one of its N_p elements as the operating point of the robot. TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) [57] is a well-established method for this kind of problem: it ranks the candidates by how close they are to a hypothetical *ideal* solution (best value on every criterion) and how far they are from the *anti-ideal* solution (worst value on every criterion).

Each criterion is first normalised through

$$\bar{F}_{ij} = \frac{F_{ij}}{\sqrt{\sum_{k=1}^{N_p} F_{kj}^2}}, \quad (26)$$

and weighted by a user-supplied importance vector $\mathbf{w} = [w_1, \dots, w_m]^\top$ with $\sum_j w_j = 1$, producing the weighted normalised matrix $V_{ij} = w_j \bar{F}_{ij}$. The ideal and anti-ideal solutions are defined componentwise as

$$V_j^+ = \max_i V_{ij}, \quad V_j^- = \min_i V_{ij}, \quad (27)$$

under the convention that all objectives are to be maximised; the Euclidean distance of each candidate to the ideal and anti-ideal solutions is

$$d_i^\pm = \sqrt{\sum_{j=1}^m (V_{ij} - V_j^\pm)^2}, \quad (28)$$

and the closeness coefficient

$$C_i = \frac{d_i^-}{d_i^+ + d_i^-} \in [0, 1] \quad (29)$$

ranks the candidates: $C_i = 1$ corresponds to the ideal solution and $C_i = 0$ to the anti-ideal one. The Pareto-optimal policy associated with the highest C_i is returned as the recommended operating point of the framework. Because the TOPSIS step operates on the relatively small front $\mathcal{P}$ rather than on the (much larger) raw output of the RL procedure, it is cheap to evaluate and can be re-run with different importance vectors $\mathbf{w}$ without retraining the policy.

4.4. Extension to six-degree-of-freedom task-space motions

For brevity, the developments of Sections 4.1–4.3 were stated for a one-dimensional motion $y(t)$, and the case study of Section 5.4 models a single Cartesian coordinate as a DMP while the orthogonal coordinates follow pre-defined motions. The framework, however, extends naturally to the full six-degree-of-freedom (6-DOF) end-effector pose, that is to a translation $\mathbf{p} \in \mathbb{R}^3$ and to an orientation typically represented either as Euler (roll–pitch–yaw) angles $\boldsymbol{\eta} \in \mathbb{R}^3$ or as a unit quaternion $\mathbf{q} \in S^3$. Orientation is not a Euclidean quantity, and a body of work accordingly learns skills directly on the corresponding manifolds — Saveriano et al. [62], for instance, learn stable dynamical systems on Riemannian manifolds so that unit-quaternion and symmetric-positive-definite constraints are satisfied by construction. The construction below summarises the extension and discusses the orientation-specific considerations; the implementation of the experimental section can be lifted to 6-DOF essentially without change.

Translational DMP. The three Cartesian components of the end-effector position $\mathbf{p} = [p_x, p_y, p_z]^\top$ are encoded by three transformation systems that share the same canonical phase $x(t)$,

$$\tau \dot{\mathbf{z}}_p = \mathbf{K}_p(\mathbf{g}_p - \mathbf{p}) - \mathbf{D}_p \mathbf{z}_p - \mathbf{K}_p(\mathbf{g}_p - \mathbf{p}_0)x + \mathbf{K}_p \mathbf{f}_p, \quad \tau \dot{\mathbf{p}} = \mathbf{z}_p, \quad (30)$$

in which $\mathbf{K}_p, \mathbf{D}_p \in \mathbb{R}^{3 \times 3}$ are diagonal stiffness and damping matrices satisfying the critical-damping condition componentwise, and $\mathbf{f}_p \in \mathbb{R}^3$ is the vector of forcing terms emitted by the policy network. Because the three transformation systems share the same phase, the per-component forcing term retains the form (4), but the basis-function weights, and hence the deep policy of Section 4.1.1, are independent across the three axes. The state-space enrichment (12) becomes

$$\mathbf{s}_t = (x_t, \mathbf{p}_t, \dot{\mathbf{p}}_t) \in \mathbb{R}^7, \quad (31)$$

and the policy outputs a three-dimensional forcing vector $\mathbf{f}_t = \mathbf{h}_\theta(\mathbf{s}_t) x_t \in \mathbb{R}^3$. The spatial-scaling transformations (17) extend componentwise with a diagonal scaling matrix $\mathbf{A} = \text{diag}(A_x, A_y, A_z)$, where $A_i = (g'_i - p'_{0,i})/(g_i - p_{0,i})$ for $i \in \{x, y, z\}$.

Orientation DMP via roll–pitch–yaw. When the working range of the end-effector is small enough to remain bounded away from kinematic singularities of the Euler-angle parameterisation, the orientation can be encoded by three additional scalar DMPs on (η_r, η_p, η_y) that follow exactly the same structure as (30)–(31) with a state vector $\mathbf{s}_t^\eta = (x_t, \boldsymbol{\eta}_t, \dot{\boldsymbol{\eta}}_t)$.

The total state of the 6-DOF policy is then the concatenation $\mathbf{s}_t^{\text{6D}} = (x_t, \mathbf{p}_t, \dot{\mathbf{p}}_t, \boldsymbol{\eta}_t, \dot{\boldsymbol{\eta}}_t) \in \mathbb{R}^{13}$ and the action is the six-dimensional forcing $\mathbf{f}_t^{\text{6D}} \in \mathbb{R}^6$. The same supervised pre-training, multi-objective refinement and TOPSIS selection apply unchanged.

Orientation DMP via unit quaternions.. For motions that span a wider orientation range or that traverse a singularity of the Euler-angle parameterisation, the orientation should be encoded on the unit-quaternion manifold S^3 rather than in $\mathbb{R}^3$. Following the formulation of Saveriano et al. [6], a quaternion DMP replaces the linear position error $(\mathbf{g}_p - \mathbf{p})$ with the logarithmic map of the relative quaternion,

$$\tau \dot{\omega} = \mathbf{K}_o 2 \log(\mathbf{q}_g * \bar{\mathbf{q}}) - \mathbf{D}_o \omega - \mathbf{K}_o 2 \log(\mathbf{q}_g * \bar{\mathbf{q}}_0) x + \mathbf{K}_o \mathbf{f}_o, \quad (32)$$

together with the quaternion-product integration $\dot{\mathbf{q}} = \frac{1}{2} \omega * \mathbf{q}$, where $\omega \in \mathbb{R}^3$ is the body angular velocity, $\mathbf{q}_g, \mathbf{q}_0, \mathbf{q} \in S^3$ are the goal, start and current quaternions, $\bar{\mathbf{q}}$ denotes the quaternion conjugate, $*$ denotes Hamilton’s product, $\log(\cdot)$ is the quaternion logarithm and $\mathbf{f}_o \in \mathbb{R}^3$ is the orientation forcing term. The deep policy is then parameterised on the augmented state $(x_t, \mathbf{q}_t, \omega_t)$ with an output projected on the tangent space at the current quaternion, and the spatial-scaling argument of Section 4.1.3 applies with the appropriate quaternion-logarithm ratio. The multi-objective machinery and the TOPSIS step are unaffected by the change of orientation parameterisation, which only modifies the integration of the transformation system and the input dimension of the policy network.

The 6-DOF extension introduces three additional considerations that are worth highlighting. First, although the canonical system is shared across the six channels, the per-channel policies need not be: independent networks decouple the per-axis convergence rates at the cost of a slightly larger total parameter count. Second, the inverse-kinematics block of Fig. 1 now consumes a full pose reference rather than a single Cartesian coordinate; for the manipulator, an analytical closed-form solver [63] is available and avoids the introduction of numerical conditioning issues. Third, when orientation is represented by quaternions, the imitation reward (21) and the velocity reward (22) should be computed with the geodesic distance on S^3 and with the angular-velocity norm respectively, so that the vector reward retains its physical interpretation. The experimental scope of this extension is limited. The multi-axis *position* construction (30)–(31) is exercised empirically in Section 5.12, where three parallel position DMPs sharing one canonical system are fitted and redeployed at a new start–goal pair; the orientation extensions of this subsection (Euler-angle and quaternion), by contrast, are derived but *not* yet validated experimentally. The main case study therefore restricts the learnt DMP to a single Cartesian component to keep the multi-objective trade-off easy to visualise and compare, and a full 6-DOF pose experiment — in particular validating the quaternion forcing term and the geodesic rewards on real orientation data — is left to future work.

4.5. Algorithmic pseudocode

For completeness and reproducibility, Algorithms 1–3 consolidate the operations of the three stages of the framework into pseudocode. The notation follows the previous subsections. The pseudocode is back-end-agnostic: the function `POLICYGRADIENTUPDATE` in Algorithms 1 and 2 refers to one update step of the chosen actor–critic algorithm (TD3, SAC or PPO), and `VECTORVALUEPOLICYGRADIENTUPDATE` in Algorithm 2 refers to its vector-valued multi-objective extension as described in Section 4.2.1.

Algorithm 1 Stage I — Imitative learning of the DMP policy.

Require: demonstration $\{y_{\text{demo}}(t), \dot{y}_{\text{demo}}(t), \ddot{y}_{\text{demo}}(t)\}_{t=1}^P$, hyper-parameters $(\tau, K, D, \alpha_x, N, T)$, supervised epochs E_{sup} , RL episodes E_{imit}

Ensure: trained DMP policy $h_{\theta_{\text{imit}}}^*$

- 1: Set $y_0 \leftarrow y_{\text{demo}}(1)$, $g \leftarrow y_{\text{demo}}(P)$
 - 2: Generate phase trajectory $\{x_t\}$ from (9)
 - 3: Compute supervision targets $\{h_{\text{target}}(t)\}$ from (19)
 - 4: Initialise θ randomly; build dataset $\mathcal{D}$ from (20)
 - 5: **for** $k = 1$ **to** E_{sup} **do** ▷ Supervised pre-training
 - 6: $\theta \leftarrow \theta - \eta \nabla_{\theta} \|h_{\text{target}} - h_{\theta}\|_{\mathcal{D}}^2$
 - 7: **end for**
 - 8: Initialise critic networks randomly
 - 9: **for** $e = 1$ **to** E_{imit} **do** ▷ Deep RL refinement
 - 10: Roll out an episode of length P using (9)–(10) and action $f_t = h_{\theta}(\mathbf{s}_t) x_t$
 - 11: Collect rewards $r(t)$ from (21)
 - 12: $\theta \leftarrow \text{POLICYGRADIENTUPDATE}(\theta, \{(\mathbf{s}_t, f_t, r(t))\})$
 - 13: **end for**
 - 14: **return** $h_{\theta_{\text{imit}}}^* \leftarrow h_{\theta}$
-

The pseudocode makes explicit the four parameters that the user must supply: the DMP hyper-parameters $(\tau, K, D, \alpha_x, N)$ shared by Stages I and II; the supervised and RL budgets $(E_{\text{sup}}, E_{\text{imit}})$ of Stage I; the population, warm-up, generation and update budgets $(n, E_{\text{wu}}, G, m_t, K_w)$ of Stage II; and the importance vector $\mathbf{w}$ of Stage III. Of these, only the importance vector $\mathbf{w}$ encodes *task-level* preferences and is therefore the only parameter that a non-expert user is expected to adjust after deployment; the other parameters are tuned once and reused across tasks of the same kinematic structure.

4.6. Computational complexity

This subsection summarises the computational cost of each stage of the framework in terms of the hyper-parameters introduced in Sections 4.1–4.3 and listed in Algorithms 1–3. Throughout, $|h_{\theta}|$ denotes the parameter count of the policy network, $|V|$ the parameter count of the value (or vector-value) network, P the length of an episode in time steps, d_s the state dimension, d_a the action dimension and m the number of objectives. For the case-study setting of Table 1, $P = 100$, $d_s = 3$, $d_a = 1$ and the network architecture of Section 5.1 yields $|h_{\theta}| \sim 7 \cdot 10^4$ parameters.

Algorithm 2 Stage II — Multi-objective refinement and Pareto-front construction.

Require: initial policy $h_{\theta_{\text{init}}}^*$, vector reward $\mathbf{r}(t)$ as in (22), population size n , warm-up iterations E_{wu} , generations G , per-pair updates m_t , simplex resolution K_w

Ensure: discrete Pareto front $\mathcal{P}$ of policies and objectives

```

1: Sample weight set  $\Omega = \{\omega_i\}_{i=1}^n$  on the unit simplex
2: Initialise population  $\Pi = \{\pi_i\}_{i=1}^n$  from  $h_{\theta_{\text{init}}}^*$ 
3: Initialise prediction-model buffer  $\mathcal{R} \leftarrow \emptyset$ 
4: for  $e = 1$  to  $E_{\text{wu}}$  do                                 $\triangleright$  Warm-up stage
5:   for  $i = 1$  to  $n$  do
6:     Roll out one episode under  $(\pi_i, \omega_i)$ ; collect  $\mathbf{F}(\pi_i)$ 
7:      $\pi_i \leftarrow \text{VECTORVALUEPOLICYGRADIENTUPDATE}(\pi_i, \omega_i)$ 
8:     Append  $(\mathbf{F}(\pi_i), \mathbf{F}(\pi'_i), \omega_i)$  to  $\mathcal{R}$ 
9:   end for
10: end for
11: Compute current Pareto front  $\text{EP} \leftarrow \text{PARETO}(\{\mathbf{F}(\pi_i)\})$ 
12: for  $g = 1$  to  $G$  do                                 $\triangleright$  Evolutionary stage
13:   Fit prediction model  $\Delta_j^i(\omega_j)$  from (24) on  $\mathcal{R}$ 
14:   Discretise simplex into  $K_w$  samples and solve (25) by
       knapsack reduction to obtain  $\mathcal{T} = \{(\pi_i, \omega_i)\}_{i=1}^n$ 
15:   for  $i = 1$  to  $n$  do
16:     for  $\ell = 1$  to  $m_t$  do
17:        $\pi_i \leftarrow \text{VECTORVALUEPOLICYGRADIENTUPDATE}(\pi_i, \omega_i)$ 
18:     end for
19:     Roll out one episode; append  $(\mathbf{F}(\pi_i^{\text{old}}), \mathbf{F}(\pi_i), \omega_i)$  to
        $\mathcal{R}$ 
20:   end for
21:    $\text{EP} \leftarrow \text{PARETO}(\text{EP} \cup \{\mathbf{F}(\pi_i)\}_{i=1}^n)$ 
22: end for
23:  $\mathcal{P} \leftarrow \{(\pi, \mathbf{F}(\pi)) : \mathbf{F}(\pi) \in \text{EP}\}$ 
24: return  $\mathcal{P}$ 

```

Algorithm 3 Stage III — TOPSIS operating-point selection.

Require: Pareto front $\mathcal{P} = \{(\pi_i, \mathbf{F}_i)\}_{i=1}^{N_P} \subset \mathbb{R}^m$, importance vector $\mathbf{w} \in \mathbb{R}_{\geq 0}^m$ with $\sum_j w_j = 1$

Ensure: selected policy h_{θ}^*

```

1: for  $i = 1$  to  $N_P$ ,  $j = 1$  to  $m$  do
2:    $\bar{F}_{ij} \leftarrow F_{ij} / \sqrt{\sum_{k=1}^{N_P} F_{kj}^2}$                                  $\triangleright$  Normalisation (26)
3:    $V_{ij} \leftarrow w_j \bar{F}_{ij}$ 
4: end for
5: for  $j = 1$  to  $m$  do
6:    $V_j^+ \leftarrow \max_i V_{ij}$ ,    $V_j^- \leftarrow \min_i V_{ij}$ 
7: end for
8: for  $i = 1$  to  $N_P$  do
9:    $d_i^\pm \leftarrow \sqrt{\sum_{j=1}^m (V_{ij} - V_j^\pm)^2}$ 
10:   $C_i \leftarrow d_i^- / (d_i^+ + d_i^-)$ 
11: end for
12:  $i^* \leftarrow \arg \max_i C_i$ 
13: return  $h_{\theta}^* \leftarrow \pi_{i^*}$ 

```

Stage I — supervised pre-training. The supervised target $h_{\text{target}}(t)$ in (19) is obtained in closed form from the demonstration in $O(P)$ operations. The mini-batch gradient descent of Algorithm 1 costs $\mathcal{O}(E_{\text{sup}} \cdot |h_{\theta}|)$ for the parameter updates plus $\mathcal{O}(E_{\text{sup}} \cdot P \cdot d_s)$ for the forward passes over the demonstration. With the network of Section 5.1, this stage requires only a few seconds on a single CPU.

Stage I — RL refinement. The deep RL refinement loop performs one environment step (a constant-time evaluation of (9)–(10)) followed by one network forward pass per time step. For E_{init} episodes of length P , the total cost is

$$C_{\text{Stage I, RL}} = \mathcal{O}(E_{\text{init}} \cdot P \cdot (|h_{\theta}| + |V|)), \quad (33)$$

which is independent of the dimensionality of the workspace because (a) the canonical system is shared across Cartesian coordinates and (b) per-axis DMPs are decoupled (Section 4.4). This is the empirical observation reported in Table 2: tripling the workspace dimensionality from 1 to 3 does not appreciably increase the wall-clock cost of Stage I.

Stage II — multi-objective refinement. The dominant cost of Algorithm 2 is the per-pair policy gradient. The warm-up stage performs $E_{\text{wu}} \cdot n$ episodes; the evolutionary stage performs $G \cdot n \cdot m_t$ episodes; each environment step costs $\mathcal{O}(|h_{\theta}| + m \cdot |V|)$ because the vector-valued critic carries m outputs in parallel. The total RL cost is therefore

$$C_{\text{Stage II, RL}} = \mathcal{O}(n \cdot P \cdot (|h_{\theta}| + m \cdot |V|) \cdot (E_{\text{wu}} + G \cdot m_t)). \quad (34)$$

On top of the RL cost, each generation of the evolutionary stage fits the analytical prediction model (24) on the buffer $\mathcal{R}$ at a cost $\mathcal{O}(|\mathcal{R}|)$ per (i, j) pair via Levenberg–Marquardt, and selects the next n pairs by solving the knapsack-reduced quadratic problem (25) in $O(K_w \cdot |\text{EP}|)$ time, where K_w is the simplex discretisation and $|\text{EP}|$ the size of the current front. Both terms are negligible with respect to (34) in our experiments. The continuous-Pareto-front representation of Section 4.2.3 adds an $O(N_P^2)$ t-SNE step [60] and an $O(N_P \cdot k \cdot i_k)$ k -means step, where N_P is the size of the final front, k the number of clusters and i_k the number of k -means iterations.

Stage III — TOPSIS. Algorithm 3 touches the front exactly twice and the importance vector once. Its cost is $O(N_P \cdot m)$, which is several orders of magnitude smaller than (34). Re-running Stage III with a different importance vector $\mathbf{w}$ does *not* require retraining, which is the practical advantage that justifies its inclusion as a separate stage.

Memory. The bottleneck is the replay buffer of the off-policy actor–critic algorithms (TD3, SAC), whose size is fixed and back-end-defined; the framework adds only the small prediction-model buffer $\mathcal{R}$ of triples $(\mathbf{F}(\pi), \mathbf{F}(\pi'), \omega) \in \mathbb{R}^{2m+m}$ used to fit (24). For the parameter ranges of our experiments, $|\mathcal{R}| \sim 10^2$ entries, so the additional memory footprint is negligible.

Wall-clock summary. On the workstation of Section 5.1, Stage I converges in less than two hours irrespective of the back-end (Table 2); Stage II requires $E_{\text{wu}} + G \cdot m_t \approx 100$ inner-loop iterations per policy with $n = 10$ policies in parallel and 6 rollout episodes per update (i.e. the 6000 episodes of Section 5.3), yielding a wall-clock cost approximately one order of magnitude larger than Stage I; Stage III is essentially instantaneous. Both stages are amenable to embarrassingly parallel execution across the n policies and across the random seeds, which is exploited in the ablation studies of Section 5.

4.7. Pipeline summary

The complete framework, summarised pictorially in Fig. 2, takes a single demonstration as input and returns a refined DMP policy as output, with no human intervention beyond the choice of DMP hyper-parameters (Section 4.1.1), the multi-objective deep RL hyper-parameters (Section 4.2) and the importance vector $\mathbf{w}$ that encodes the user’s preferences in Stage III. Stages II and III are decoupled by the explicit construction of the Pareto front, so that a single Stage-II run can be reused with several different TOPSIS preference vectors at virtually no additional cost. Likewise, the deep RL back-end used in Stages I and II is interchangeable: as discussed in Section 3.2, any policy-gradient algorithm with continuous state and action spaces is admissible. The next section presents an empirical evaluation of the framework with three popular back-ends (TD3, SAC and PPO) on canonical reference trajectories and on a pick-and-place case study in MuJoCo.

5. Experiments and Results

This section reports an empirical evaluation of the proposed framework. Section 5.1 describes the implementation and the hyper-parameter choices that are common to all experiments. Section 5.2 evaluates the Stage I imitation pipeline on canonical sine and cosine trajectories with three deep RL back-ends (TD3, SAC, PPO) and compares them on accuracy and computational cost. Section 5.3 presents Stage II results, namely the Pareto front of imitation versus motion effort on a sine reference. Section 5.4 presents a pick-and-place case study performed in MuJoCo that exercises the complete three-stage pipeline, Section 5.5 validates the same skill on a physical manipulator through a 250-trial hardware experiment, Section 5.6 reports a 360-trial cross-condition robustness study on the physical robot, and Section 5.7 reports a 240-trial study of the actuator effort, smoothness and dynamic load. The remaining subsections report ablation, sensitivity, generalisation, three-objective and obstacle-avoidance studies.

5.1. Implementation and Hyper-parameter Settings

All experiments were implemented in Python using PyTorch and standard open-source implementations of the three deep RL back-ends. The MuJoCo physics engine [64] was used for the rigid-body simulation of the manipulator. Training and inference were performed on a workstation equipped with a 7-core Intel® CPU at 2.59 GHz, an NVIDIA GeForce

940MX GPU together with an integrated Intel HD Graphics 520, and 8 GB of RAM. The DMP hyper-parameters used throughout the paper are listed in Table 1; these were chosen with $D = 2\sqrt{K}$ so that the goal attractor of (1)–(2) is critically damped. The DMP forcing term (4) uses N Gaussian basis functions whose centres are spaced exponentially in the phase variable, following the time-uniform placement rule of [8] (with a centre-spacing decay constant larger than the canonical rate α_x , so that the bases concentrate towards the end of the phase decay), and whose widths are proportional to the local centre spacing.

On the nature of the demonstrations. A scoping remark applies to all experiments below: the demonstrations used here are prescribed *analytic reference trajectories* rather than human-recorded motions. Three points bound what this does and does not affect.

First, the demonstration enters the framework at exactly two places: as the sampled target sequence in the supervised pairs (20) that bootstrap the policy in Stage I, and as the target of the imitation reward (21). Both are obtained from the sampled triple $\{y(t), \dot{y}(t), \ddot{y}(t)\}$ by the closed-form fit (7), which is indifferent to how those samples were produced. Nothing in Stages I–III inspects the provenance of the reference, so a kinesthetic, teleoperated or marker-tracked trajectory would enter through the same interface with no change to any algorithm, hyper-parameter or reward.

Second, and more substantively, the reference does not determine the refined behaviour. Stage I is a *bootstrap*: its role is to place the policy near a good initialisation, and its influence is measured directly in the ablation of Section 5.8, where removing it degrades the final return but does not change the character of the solution. The multi-objective refinement of Stage II optimises the vector reward (22), whose effort component is evaluated on the executed motion, and the operating point is then selected from the resulting Pareto front. The demonstration therefore fixes the imitation *target* and the starting point of the search; it does not fix the trade-off that the framework discovers, nor the policy that TOPSIS selects.

Third, the choice is deliberate rather than incidental. Human demonstrations differ between repetitions, so an evaluation built on them cannot separate variation caused by the refinement stages from variation already present in the input. Controlled references remove that confound: every imitation residual, Pareto front and effort reduction reported below is attributable to the algorithm alone and is exactly reproducible. This matters here because the claims are comparative — five controllers on the same task, one front against its own extremes — and such comparisons are only meaningful if the input is held fixed.

What remains untested is bounded and specific. Recorded human motion carries properties that smooth analytic curves lack: correlated sensor noise, non-smooth segments and hesitations, via-point structure, and inter-demonstration variability that forces a choice between fitting one trajectory or an average of several. The evidence in Section 5.13 and Table 9 shows the fitting stage degrading gracefully when the basis is too coarse

Table 1: DMP hyper-parameters used in all experiments.

Parameter	Value
Number of integration steps (per episode)	100
Motion duration τ	1 s
Stiffness gain K	1000
Damping gain D	63.25
Canonical decay rate α_x	1
Integration step T	0.01 s
Number of basis functions N	25

to represent the target, and the hardware experiments show the refinement operating under real execution noise, friction and payload variation; neither establishes that the *fit* of a noisy, irregular reference behaves as the fit of a smooth one. Validating the pipeline on physically taught demonstrations is accordingly identified as the most direct item of future work (Section 6). The results below should be read as a characterisation of the framework on controlled references, and the framing throughout is one of trajectory refinement rather than demonstration acquisition.

For the three deep RL back-ends, both the actor and the critic networks are fully connected feed-forward networks with two hidden layers of 256 units each and rectified-linear activations, following [11, 10]. The learning rate of all Adam optimisers is 3×10^{-4} . For TD3, the exploration noise is a zero-mean Gaussian with standard deviation 0.1. Hyper-parameters that are specific to each back-end (replay-buffer size, batch size, target-update rate, clip ratio, GAE parameters) are kept at the default values reported in the original papers; the few that matter for the comparison reported below (e.g. the number of environment steps per gradient update) are stated explicitly when needed.

5.2. Stage I: Imitation Learning on Canonical Trajectories

To evaluate the proposed DMP-as-policy formulation in a controlled setting, the framework was first asked to learn two canonical reference trajectories of duration 1 s,

$$y_{\text{demo}}(t) = \cos(\pi t), \quad y_{\text{demo}}(t) = \sin(2\pi t), \quad (35)$$

both sampled with the integration step T of Table 1. The first trajectory is a half period of a cosine that starts at 1 and ends at -1 with non-zero start and end velocity; the second is a full period of a sine that starts and ends at zero, which would induce a singularity in the original Ijspeert formulation but is handled correctly by the Park-variant DMP (1)–(2) thanks to its additional $-K(g - y_0)x$ term. After supervised pre-training as described in Section 4.1.4, the policy was refined with each of TD3, SAC and PPO under the imitation reward (21).

Fig. 3 (top) shows the trajectory reproduced by the TD3 back-end after convergence on the cosine target, superimposed on the demonstration. The reproduced trajectory tracks the demonstration with a small, bounded residual that is consistent with the limited capacity of the $N = 25$ basis-function expansion. The corresponding training curve (bottom) shows a

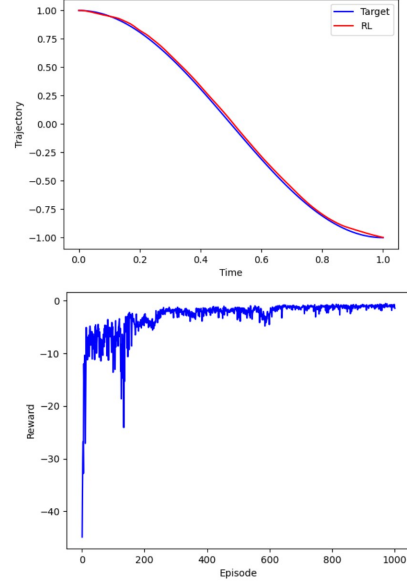


Figure 3: Imitation results on the cosine reference using TD3 as the deep RL back-end. (Top) After training, the learnt DMP-as-policy (red) reproduces the demonstration (blue) with a small bounded residual. (Bottom) Episodic return as a function of training episode, demonstrating monotone convergence to a near-zero return.

Table 2: Comparison of the three deep RL back-ends on the canonical-trajectory imitation task. Reported values are the mean number of episodes required for the return to plateau and the corresponding wall-clock training time on the workstation described in Section 5.1.

Algorithm	Episodes to convergence	Wall-clock time (min)
PPO [9]	4000	26
SAC [10]	1000	86
TD3 [11]	1500	100

typical actor–critic learning dynamics, with the episodic return rising from approximately -45 at initialisation to a near-zero saturation value after roughly 400 episodes. Similar curves were obtained for SAC and PPO and on the sine target, with the differences across algorithms summarised quantitatively in Table 2.

Although SAC reaches convergence in the fewest episodes, the per-episode cost of its off-policy actor–critic updates makes it slower in wall-clock time than the on-policy PPO. TD3 lies between SAC and PPO in number of episodes but is the slowest in wall-clock time because of the cost of its twin critics and target-policy smoothing. The DMP-as-policy formulation does not introduce any structural bias that favours one algorithm, so this ordering is consistent with previous reports on standard MuJoCo benchmarks [11, 10]. Because the multi-objective stage of Section 4.2 requires a relatively large number of independent policy-gradient runs and is therefore dominated by the cost of the back-end, PPO is adopted as the back-end for the remainder of the experiments. It is also worth noting that the DMP-as-policy formulation models each Cartesian compo-

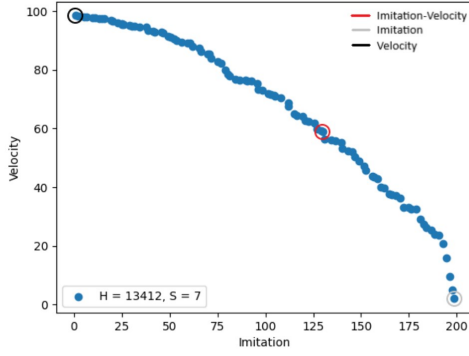


Figure 4: Discrete Pareto front obtained by Stage II on the sine reference trajectory $y_{\text{demo}}(t) = \sin(2\pi t)$. Each blue marker corresponds to one Pareto-optimal policy in the imitation-versus-velocity objective space. The three coloured markers indicate the imitation-only solution (grey), the velocity-only solution (black), and the balanced TOPSIS-selected solution (red). Hypervolume $H = 13412$, sparsity indicator $S = 7$ (smaller is denser).

ment of the end-effector trajectory as an independent DMP that shares the same canonical system, so the dimensionality of the workspace does not affect the cost reported in Table 2.

5.3. Stage II: Multi-objective Refinement on the Sine Reference

To exercise the multi-objective refinement procedure of Section 4.2, the policy obtained in Section 5.2 for the sine reference was refined against the vector reward (22). Ten policy-weight pairs $\{(\pi_i, \omega_i)\}_{i=1}^{10}$ were updated in parallel for a total of 6000 training episodes, with 40 warm-up iterations during which the population is initialised, $G = 6$ generations of the evolutionary stage, $m_t = 10$ multi-objective policy-gradient updates per generation and 6 rollout episodes per update (so that $n \cdot (E_{\text{wu}} + G m_t) \cdot 6 = 6000$). The weight vectors were sampled from the unit simplex with each component bounded to $[0, 1]$.

Fig. 4 reports the resulting discrete Pareto front in objective space; the horizontal axis is the imitation return (negative absolute deviation between the learnt and demonstrated trajectories, integrated over the episode) and the vertical axis is the velocity penalty (negative absolute end-effector velocity, integrated over the episode), both rescaled to a $[0, 200] \times [0, 100]$ range to ease visualisation. Each blue marker corresponds to one Pareto-optimal policy. The three coloured circles identify three operating points of particular interest: the IMITATION-only solution (grey) that maximises the imitation return at the cost of high velocity, the VELOCITY-only solution (black) that minimises the velocity penalty at the cost of poor imitation, and a balanced IMITATION-VELOCITY trade-off (red) returned by TOPSIS with equal weights on the two criteria. The hypervolume of the front is $H = 13412$ and the sparsity indicator is $S = 7$, the large dominated region and the small inter-point gaps together indicating a well-converged and densely sampled Pareto-optimal set. At the imitation-only extreme, the imitation return is 198.77 and the velocity penalty is 2.007, illustrating the antagonistic nature of the two objectives that justifies the multi-objective treatment.

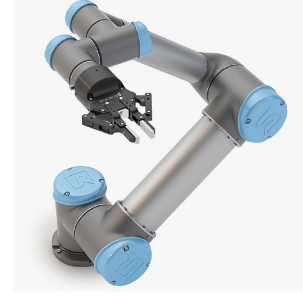


Figure 5: The six-degree-of-freedom collaborative manipulator with a two-finger gripper attached at the end-effector [63], as modelled in the case study of Section 5.4. The physical cell used in Experiments 1–3, together with its digital twin, is shown in Fig. 8.

5.4. Case Study: Pick-and-Place in Simulation

The complete pipeline was finally evaluated on a pick-and-place task performed by a six-degree-of-freedom collaborative manipulator equipped with a two-finger gripper [63], simulated in MuJoCo. The manipulator is shown in Fig. 5; in the simulated cell the robot must pick a wooden block on its right-hand-side table and place it on a target table to its left. The orientation of the end-effector is held downward throughout the motion. The action sequence consists of four phases: an initial reaching motion to the pickup location, a grasp by the gripper, a lifting motion, and a final transport motion to the placement location. The DMP framework is used to model the final transport motion only; the three earlier phases are pre-scripted and identical across all experiments.

A single demonstration of the transport motion was provided in task space. The demonstration constrains only the p_y component of the end-effector position; the orthogonal components are kept on a pre-defined straight segment. Writing the Cartesian end-effector position as $\mathbf{p} = [p_x, p_y, p_z]^T$ (distinct from the canonical phase x of Section 3.1), the demonstration is

$$p_y(t) = 0.8 \sin\left(\frac{2\pi}{3} t\right), \quad p_x(t) = 0.6(t - 1), \quad p_z(t) = 1, \quad (36)$$

with $t \in [0, 1]$ s and all coordinates expressed in metres. The p_y component is modelled by a single DMP with the hyperparameters of Table 1 and learnt by the Stage I pipeline using PPO. At run time the desired task-space motion is converted to joint commands by an analytical inverse-kinematics solver [63, 65]; of the (up to eight) inverse-kinematics branches, the one closest to the current joint configuration is selected (elbow-up, non-flipped wrist), which keeps the arm on a single smooth branch throughout the transport and avoids reconfiguration jumps. The resulting joint references are tracked by the proportional-derivative (PD) joint controller of Table 3 augmented with a gravity-compensation term $\hat{\mathbf{g}}(\mathbf{q})$ (computed from the model link masses and centres of mass), so that the modest PD gains in Table 3 (in N m/rad and N m s/rad) regulate only the comparatively small dynamic tracking error rather than the standing gravity load; this is why gains of order 15–30 N m/rad suffice despite a gravity-induced shoulder torque of tens of newton-metres.

Table 3: Joint and gripper PD-controller gains used in the case study, tuned manually on the MuJoCo model. K_p is in N m/rad and K_d in N m s/rad; the controller additionally applies a model-based gravity-compensation term, so these gains regulate only the dynamic tracking error.

Joint	K_p	K_d
1	21	0.11
2	30	0.10
3	15	0.05
4	21	0.01
5	15	0.01
6	15	0.01
Gripper	7.5	0

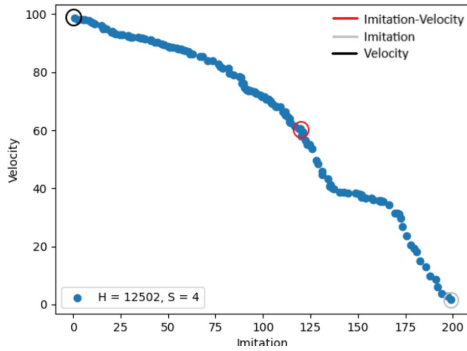


Figure 6: Discrete Pareto front obtained by Stage II on the pick-and-place case study. Axes and markers are as in Fig. 4. Hypervolume $H = 12502$, sparsity indicator $S = 4$ (smaller is denser).

The Stage II multi-objective refinement was then run on the trained primitive using the same vector reward (22) and the same population size and budget as in Section 5.3. Fig. 6 reports the resulting discrete Pareto front. The hypervolume is $H = 12502$ and the sparsity indicator is $S = 4$, confirming the quality and the dense sampling of the front. As in the canonical-trajectory case, the front exhibits a clear trade-off between imitation fidelity and velocity penalty.

To visualise the qualitative effect of selecting different operating points on the front, Fig. 7 shows the end-effector trajectories, viewed from above the workspace, produced by two extremal Pareto-optimal policies. Maximising the imitation reward alone (left) produces a curved trajectory that follows the sinusoidal shape of the demonstration; maximising the velocity reward alone (right) produces an almost straight diagonal between the pickup and the placement locations, which has minimal length but completely discards the demonstration’s shape. The balanced TOPSIS solution lies between these two extremes; it preserves the sinusoidal trend of the demonstration while reducing the integrated velocity by a factor of approximately 40% with respect to the imitation-only policy. This is consistent with the original motivation of the framework: rather than choosing a single scalar reward in advance, the user inspects the Pareto front *a posteriori* and selects an operating point that best matches the requirements of the deployed

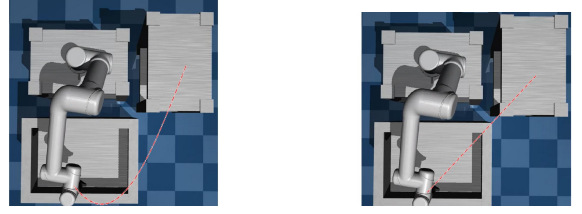


Figure 7: End-effector path of the pick-and-place task, viewed from above the workspace, for two extremal points on the Pareto front of Fig. 6. (Left) Maximising the imitation reward alone reproduces the sinusoidal shape of the demonstration. (Right) Maximising the velocity reward alone produces a straight, near-minimum-length trajectory that discards the demonstration shape. A balanced operating point selected by TOPSIS produces a trajectory that retains the sinusoidal trend while substantially reducing the integrated velocity.

system. A full per-controller quantitative comparison of these operating points — success, placement, tracking error, effort and torque — is deferred to the physical-robot experiment of Section 5.5 (Table 4), where the same three operating points are measured on hardware and the simulated $\sim 40\%$ velocity reduction is recovered as a measured 39.2%; the simulation front itself is characterised here by its hypervolume and sparsity indicator ($H = 12502$, $S = 4$, computed on the rescaled $[0, 200] \times [0, 100]$ objective axes defined in Section 3.3).

5.5. Experimental Validation on a Physical Manipulator

The simulation case study of Section 5.4 establishes the behaviour of the complete pipeline in an idealised rigid-body environment. To verify that the conclusions drawn there survive the transition to real hardware — where joint friction, payload variation, grasp uncertainty, sensing noise and finite control bandwidth are all present — the same pick-and-place task was re-instantiated on a physical collaborative manipulator and its digital twin (Fig. 8). The physical cell differs from the idealised simulation of Section 5.4 in layout and payload geometry — a single instrumented workbench carrying fiducial-marked PICK and PLACE stations and a cylindrical payload, rather than the two-table scene and prismatic block simulated in Section 5.4 — so what carries over is the three-stage pipeline and the refined transport motion, not a transplanted scene. This experiment (denoted *Experiment 1*) directly addresses the experimental-validation item raised in the conclusion of the simulation study and provides the end-to-end mechatronic evidence expected of the framework.

Controller naming and the effort objective. The three hardware experiments compare overlapping sets of controllers, referred to throughout by descriptive names rather than by per-table indices: CLASSICAL (the batch-fit DMP), SCALAR-RL (a DMP refined with a single manually weighted scalar reward), MORL-IMIT. (the imitation-fidelity extreme of the Pareto front), MORL-VEL./MORL-EFF. (the low-effort extreme), MORL-TOPSIS (the balanced point returned by the selection step), and, in Experiment 2 only, MORL-TOPSIS-CC (the same balanced policy retrained with condition coverage). MORL-TOPSIS denotes the *same* balanced operating point in

every experiment, re-derived on that experiment’s front; any M# labels in the tables are local indices. One terminological subtlety must be flagged explicitly: the second Pareto objective is the integrated end-effector *speed* $\int |\dot{\mathbf{y}}| dt$ in Stage II and in Experiments 1–2, but the *control effort* $\int \|\ddot{\mathbf{q}}\|^2 dt$ (squared joint acceleration) in the dedicated effort study of Experiment 3 (Section 5.7). Both are “low-effort” criteria, but they are not identical, and the low-effort extreme is named MORL-VEL. or MORL-EFF. accordingly; Experiment 3 quantifies precisely how speed minimisation and control-effort minimisation relate on the physical arm.

Setup and acquisition.. The experimental cell and its MuJoCo digital twin are shown in Fig. 8. The manipulator is a single laboratory manipulator unit — a six-degree-of-freedom collaborative manipulator (Denavit–Hartenberg parameters $d_1 = 0.08916$, $a_2 = -0.425$, $a_3 = -0.39225$, $d_4 = 0.10915$, $d_5 = 0.09465$, $d_6 = 0.0823$ m) running its standard industrial controller — fitted with a two-finger parallel gripper instrumented for grasp force and a wrist-mounted six-axis force/torque sensor (Fig. 8(b)). Object poses are obtained from fiducial markers on the two work stations (Fig. 8(a,b)), and a 50 mm scale bar is fixed in the workspace. This is the same physical unit used in all three hardware experiments (Sections 5.5–5.7); its dynamic parameters (link masses and centres of mass, joint viscous and Coulomb friction, motor torque constants and gripper mass) were identified once by excitation trajectories and least squares and reused across the experiments. Synchronous telemetry was acquired through the robot’s real-time data-exchange interface and logged at 100 Hz; the low-level joint controller runs on a ~ 10 ms cycle and the learnt policy is queried on a ~ 20 ms cycle. Each record contains the commanded and measured task-space pose, the measured TCP linear velocity, acceleration and jerk, the per-joint angle, velocity, torque and motor current, the wrist wrench, the gripper width/force, and the contact and safety flags; joint torque is obtained from the sensed motor current through the calibrated torque constant ($\tau = k_t I$), and the dynamic component reported later is the part above the gravity-holding baseline. As in the case study, the joint controller is a PD law (Table 3) with model-based gravity compensation, and at each step the analytical inverse-kinematics branch closest to the current configuration is selected. Measured joint angles and logged Cartesian poses were cross-checked for forward-kinematics consistency on every trial: the nominal-DH forward kinematics reproduces the logged TCP pose to within 0.72 mm in the worst trial (mean 0.57 mm), the expected kinematic-calibration residual between the nominal model and the controller’s factory-calibrated kinematics. The transport motion is the *same* reference trajectory used in simulation (36); the TCP orientation is held fixed throughout (roll = π , pitch = 0, yaw = $-\pi/2$).

Protocol.. Five controllers were compared, each corresponding to one operating point of the proposed pipeline: the classical batch-fit DMP of (7) (CLASSICAL), a DMP refined with a single manually weighted scalar reward (SCALAR-RL, whose effort weight was fixed for all trials at the best value found by

a coarse pilot sweep of the scalarisation coefficient λ in (37)), the imitation-fidelity extreme of the Pareto front (MORL-IMIT.), the TOPSIS-selected balanced policy of Section 4.3 (MORL-TOPSIS, the recommended operating point of the framework), and the velocity/effort extreme of the front (MORL-VEL.). Each controller was executed for 50 trials, giving 250 trials in total (176,106 logged samples), spread over 20 scenarios that vary the start/goal poses, the payload (0.15–0.34 kg), the support-surface friction (0.30–0.85), the object-pose noise (0.2–1.0 mm) and the presence of a circular obstacle (3 of the 20 scenarios). Trials are *paired* across controllers through a shared scenario/run identifier so that every controller is evaluated on an identical set of conditions. A trial is counted as successful only if the object is grasped, transported and placed within the placement tolerance without triggering a protective stop; placement and tracking errors are reported on successful trials, whereas the effort metrics are reported over all trials.

Quantitative results.. Table 4 summarises the per-controller outcome. The central observation is that the simulation conclusion transfers to hardware essentially unchanged: the TOPSIS-balanced policy reduces the integrated transport speed (the transport-phase path length, $\int |\dot{\mathbf{p}}| dt$) by 39.2% relative to the imitation extreme (measured 0.761 versus 1.252 m; 95% CI [0.753, 0.769] versus [1.190, 1.313] m), in close agreement with the $\sim 40\%$ reduction obtained in the MuJoCo case study of Section 5.4. This velocity reduction is achieved *without* sacrificing task quality: the TOPSIS policy attains the highest success rate (98%), the lowest task-space tracking RMSE (0.167 mm, against 0.219 mm for the imitation extreme and 0.352 mm for the velocity extreme) and the lowest mechanical-effort variability (coefficient of variation 0.024, roughly five times smaller than that of the other controllers), while its placement accuracy (1.66 mm) is statistically indistinguishable from that of the imitation-fidelity extreme (1.63 mm; overlapping 95% intervals, [1.55, 1.78] versus [1.51, 1.75] mm). All five controllers place at the 1.6–2.1 mm scale, which is set by the fiducial object-pose sensing and hand–eye calibration residual (~ 2 mm RMS) rather than by the trajectory controller — the same sensing-limited behaviour observed under varied conditions in Section 5.6. The measured mechanical effort follows the monotone ordering IMIT. > CLASSICAL > SCALAR-RL > TOPSIS > VEL. (166.6 > 155.6 > 117.0 > 110.6 > 102.7 J) that is predicted by the imitation–effort trade-off of the framework. Here the mechanical-effort index is the trajectory integral $\int |\boldsymbol{\tau} \cdot \dot{\mathbf{q}}| dt$ of the absolute joint power over the transport. It discriminates between the controllers in this experiment precisely because the five operating points execute transport paths of *different length* — the speed-minimising policies take shorter, straighter paths (0.76 m for MORL-TOPSIS against 1.25 m for MORL-IMIT.; cf. the integrated-speed row of Table 4) — so the gravity-dominated joint work integrates over a different path length for each. This is fully consistent with the controlled effort study of Experiment 3 (Section 5.7), in which the macro path is instead held *fixed* across controllers: there the bulk mechanical energy is found to be comparable (the gravity work is path- and payload-set), and the policy differences are carried

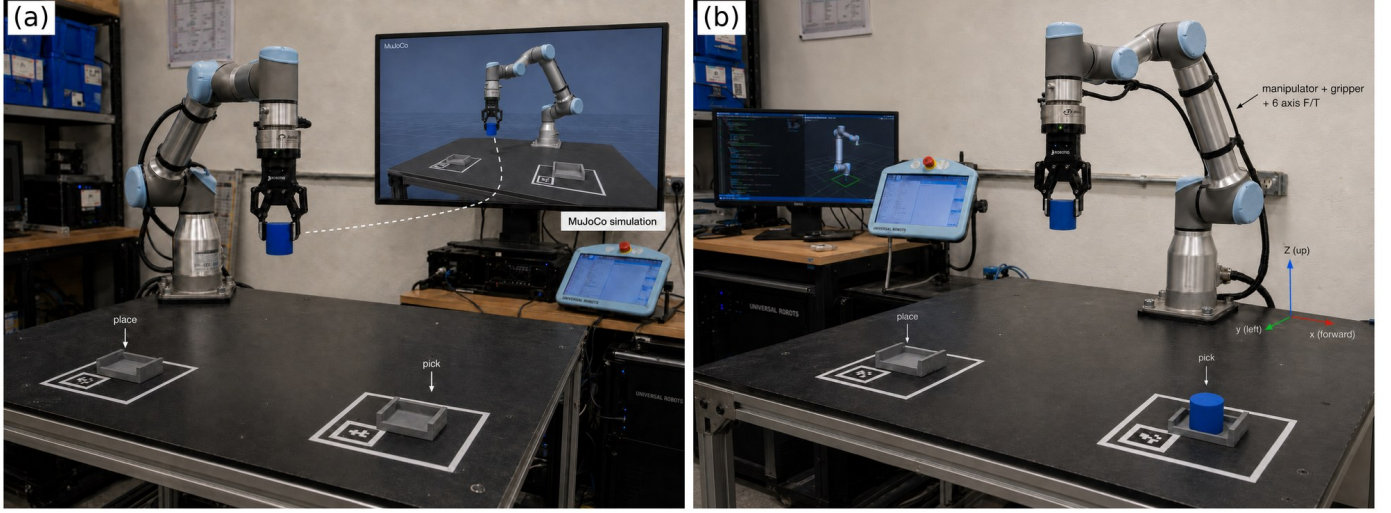


Figure 8: Physical experimental apparatus and its digital twin. (a) The manipulator hardware holding the cylindrical payload (left) and the MuJoCo digital twin of the same work cell displayed on the monitor (right); the dashed line marks the corresponding end-effector pose, making the sim-to-real correspondence explicit. The twin reproduces the layout used in the hardware experiments and is a separate model from the idealised two-table cell simulated in Section 5.4. (b) The work cell in its nominal configuration: the instrumented end-effector (two-finger parallel gripper mounted on a wrist-mounted six-axis force/torque sensor), the right-handed task frame (x forward, y left, z up) and the fiducial-marked pick and PLACE stations, with the payload seated in a 3-D-printed fixture at the pick station. The start/goal, table-height and obstacle variations of Experiment 2 are applied to this configuration.

by the dynamic, motion-shape quantities (control effort, jerk, above-gravity torque/current and vibration) rather than by bulk energy.

Qualitative behaviour. Fig. 9 reports a representative paired trial (nominal scenario, identical run identifier across all five controllers). The top-view executed TCP paths (left) reproduce, on real hardware, exactly the trade-off illustrated in simulation in Fig. 7: the imitation extreme and the classical DMP trace a pronounced curved arc that faithfully follows the demonstrated sinusoidal shape (executed transport length 1.29 and 1.24 m, peak lateral deviation 229 and 219 mm from the straight pick-to-place chord), the velocity extreme collapses onto a near-straight minimum-length path (0.75 m, 0.4 mm lateral deviation), and the TOPSIS solution sits between the two (0.76 m, 52 mm deviation), retaining a mild sinusoidal trend while shortening the path. The measured TCP speed profiles (right) make the effort difference explicit: the imitation-following controllers exhibit the characteristic twin-peak speed signature of the sinusoidal reference (the TCP decelerates through the mid-stroke inflection), the scalar-RL controller sustains the highest peak speed (0.65 m/s), and the TOPSIS and velocity controllers produce a single, lower, smoother speed hump that integrates to a substantially smaller transport velocity.

Measured Pareto front and operating-point selection. Sweeping the velocity preference on hardware yields the empirical Pareto front of Fig. 10, on which 72 of the evaluated candidate policies are non-dominated. The five controllers of Table 4 are highlighted along the front: the classical and imitation policies cluster near the low-imitation-loss/high-velocity corner, the velocity extreme occupies the opposite corner, and the TOPSIS-selected policy lands squarely on the knee of the

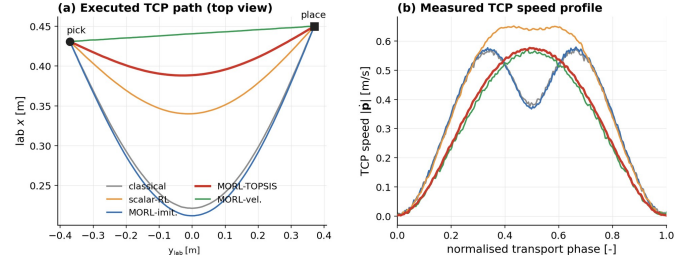


Figure 9: Measured behaviour on the physical manipulator for one paired trial (same scenario and run identifier across controllers). (a) Top-view executed TCP path: the imitation/classical controllers reproduce the curved sinusoidal demonstration, the velocity extreme follows a near-straight minimum-length path, and the TOPSIS-balanced policy lies between them. (b) Measured TCP speed magnitude over the (normalised) transport phase: the imitation-following controllers show the twin-peak signature of the sinusoidal reference, whereas TOPSIS and the velocity extreme produce a smoother, lower-velocity profile. The traces are raw 100 Hz telemetry.

measured front, the region that a human designer would intuitively choose, without a manual scalar-reward search (the objectives are first normalised onto a common scale, Section 4.3). Notably, the manually tuned SCALAR-RL controller lands at a strictly less favourable point than MORL-TOPSIS: it achieves a smaller velocity reduction (33.7% versus 39.2%), a higher tracking RMSE (0.242 versus 0.167 mm), a lower success rate (92% versus 98%) and a higher mechanical effort (117.0 versus 110.6 J), and it required a manual scalarisation weight to obtain even this result. The hardware data therefore supports the paper’s main claim: the TOPSIS selection, applied after a one-off objective normalisation, recovers, and on these metrics slightly improves upon, the operating point that scalar reward engineering reaches only after a per-task manual search.

Table 4: Physical-manipulator results (Experiment 1): five controllers, 50 trials each (250 total). Placement, tracking and grasp metrics are reported on successful trials; velocity, effort and torque metrics over all trials. Values are means; the best entry in each row is in bold (lower is better except for success rate, grasp success and speed reduction). The speed-reduction row is reported relative to the MORL-IMIT. extreme and is not bolded because a larger reduction is desirable only insofar as task quality is preserved (see text).

Metric	CLASSICAL	SCALAR-RL	MORL-IMIT.	MORL-TOPSIS	MORL-VEL.
Success rate [%]	88	92	94	98	80
Grasp success [%]	96	98	96	100	92
Failures (of 50)	6	4	3	1	10
Placement error [mm]	1.96	1.84	1.63	1.66	2.11
Tracking RMSE [mm]	0.321	0.242	0.219	0.167	0.352
Transport-phase tracking RMSE [mm]	0.349	0.272	0.237	0.176	0.367
Integrated transport speed [m]	1.154	0.830	1.252	0.761	0.684
Speed reduction vs IMIT. [%]	7.8	33.7	0.0	39.2	45.3
Transport peak speed [m/s]	0.582	0.647	0.581	0.584	0.543
Mechanical-effort index [J]	155.6	117.0	166.6	110.6	102.7
Mechanical-effort CV [-]	0.154	0.113	0.128	0.024	0.174
Peak [torque] [N m]	65.3	65.4	65.4	65.3	65.4
Control-loop query latency (incl. IK+comm.) [ms]	0.30	11.1	12.2	13.7	13.2

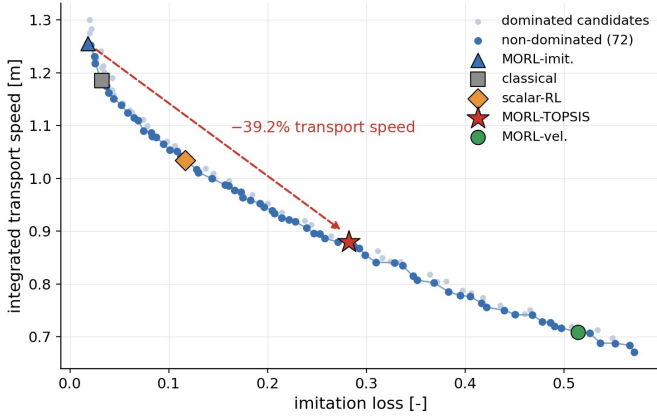


Figure 10: Measured imitation-loss-versus-transport-speed Pareto front on the physical manipulator (72 non-dominated candidate policies). The five controllers of Table 4 are marked; the TOPSIS-selected policy lies on the knee of the measured front and reduces the integrated transport speed by 39.2% relative to the imitation extreme.

The velocity extreme. The velocity extreme shows the cost of optimising one objective alone. While it minimises both the integrated transport speed (-45.3% versus the imitation extreme) and the mechanical effort (102.7 J), this aggressive minimisation degrades the metrics that matter for a deployed pick-and-place cell: its success rate falls to 80%, its tracking RMSE rises to 0.352 mm, and its failures (ten of fifty) are dominated by grasp slips (four) and protective stops (three) provoked by the higher accelerations of the shortened, straightened path. By contrast the TOPSIS policy records a single failure across all fifty trials (one placement-tolerance miss), with no grasp slips and no protective stops and a 100% grasp success rate. This is a concrete, hardware-level instance of the argument that motivates the multi-objective treatment: minimising effort in isolation is not the engineering objective, and the balanced operating point returned by the Pareto/TOPSIS stage is the one that is actually deployable. The corresponding quantitative summary,

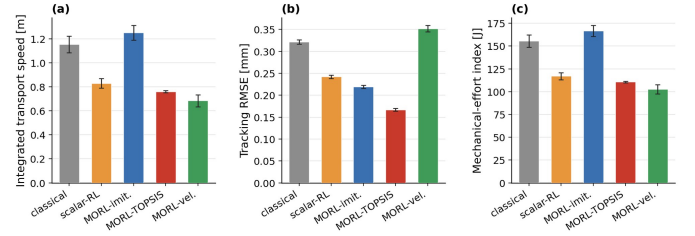


Figure 11: Per-controller hardware metrics with 95% confidence intervals ($n = 50$ trials per controller; tracking RMSE on successful trials). (a) Integrated transport speed, (b) task-space tracking RMSE and (c) mechanical-effort index. The TOPSIS-balanced policy combines a low transport speed with the lowest tracking error and a tightly concentrated effort distribution.

with 95% confidence intervals, is shown in Fig. 11.

Real-time feasibility and physical plausibility. The framework meets the timing budget of the platform. The low-level control loop held a 10.01 ± 0.16 ms period across all trials, and the control-loop query latency for the TOPSIS policy (mean 13.7 ms, 95th percentile 17.7 ms) remained below the ~ 20 ms policy-update period; the classical DMP, which requires only the analytic forcing-term integration, ran at 0.30 ms. This 11–14 ms figure is *not* the raw network evaluation (a 2×256 multilayer perceptron evaluates in well under a millisecond): it is the end-to-end control-loop query time, dominated by the analytical inverse-kinematics solve, the real-time communication round-trip and interpreter overhead, which is why the five policies report similar values despite sharing an identical network architecture. Control-deadline misses were rare (~ 1 sample per trial on average) and concentrated in the dedicated latency-stress scenario. All measured quantities stayed within the manipulator’s physical limits: the peak TCP speed never exceeded 0.70 m/s, the maximum joint velocity reached 2.45 rad/s (against the 3.14 rad/s manipulator limit), and the peak absolute joint torque was 69.3 N m, reached at the shoulder joint (joint 2), which carries the gravity load; this is well within that joint’s 150 N m rating, while the wrist joints

(joints 4–6), whose rating is 28 N m, peaked below 3.7 N m. The standing gravity-induced shoulder torque was ~ 46 N m, and the maximum motor-temperature rise over a trial was 1°C at the integer-degree resolution of the drive telemetry. Together with the sub-millimetre forward-kinematics consistency noted above, these checks confirm that the trajectories produced by the framework are not merely numerically optimal but are realised as physically valid, real-time motions on the actual robot. In summary, Experiment 1 confirms on hardware the three properties claimed in simulation — imitation of the demonstrated shape, automatic velocity/effort reduction through the Pareto/TOPSIS stage, and preservation of task accuracy — and additionally demonstrates two properties that only a physical experiment can reveal: a markedly more reliable task execution (highest success and grasp rates, fewest protective stops) and a much more repeatable effort profile for the balanced policy.

5.6. Cross-Condition Robustness on the Physical Manipulator

Experiment 1 establishes that the balanced policy transfers to hardware under a fixed nominal operating condition. A deployed manipulation skill, however, must tolerate the variability of a real cell: the start and goal poses drift, the payload and the contact friction change, the support height varies and obstacles appear. A second hardware experiment (*Experiment 2*) therefore evaluates the *cross-condition robustness* of the framework and, in particular, tests the domain-randomisation pathway anticipated in the discussion of Experiment 1: whether exposing the multi-objective refinement stage to a coverage of operating conditions yields a policy that generalises to unseen conditions without losing the imitation-fidelity and effort benefits of the method.

Controllers.. Four controllers are compared. **M1** is the classical batch-fit DMP with spatial scaling; **M2** is the scalar-weighted RL-DMP (manual reward weighting); **M3** is the proposed MORL/TOPSIS controller of Sections 4.2–4.3 trained on the nominal reference condition only (it is the same balanced operating point validated as MORL-TOPSIS in Experiment 1); and **M4** is that same MORL/TOPSIS controller trained with *condition coverage* — a domain-randomisation-style augmentation in which the multi-objective refinement is exposed to a distribution over the operating conditions and actuator/sensor parameters rather than to the nominal model alone. M4 is therefore not a new algorithm but the proposed controller equipped with the robustness augmentation; comparing M3 and M4 isolates the effect of condition coverage.

Platform and protocol.. The experiment was run on the same manipulator unit with a two-finger parallel gripper and the wrist-mounted six-axis force/torque sensor (Fig. 8), under joint PD control with gravity compensation at 100 Hz through a ROS driver; object poses were obtained from fiducial markers and fixture definitions. The unit’s dynamic parameters (link masses and centres of mass, joint viscous and Coulomb friction, motor torque constants, gripper mass) were identified beforehand by excitation trajectories and least squares, so the reported

joint torques and currents reflect the calibrated unit rather than catalogue values. Four controllers were evaluated over 30 scenarios organised into six condition groups — G1 nominal, G2 start-pose shift, G3 target-pose shift, G4 payload/friction (payload 0.12–0.35 kg, friction 0.25–0.90), G5 table-height shift and G6 obstacle — with three repetitions each, giving 360 evaluation trials. Each evaluation trial additionally draws sixteen actuator and sensing parameters from documented ranges: link mass $\pm 10\%$, link inertia $\pm 15\%$, joint viscous damping $\times [0.7, 1.4]$, Coulomb friction $\times [0.5, 1.8]$, motor torque constant $\pm 5\%$, joint-position-controller gains $K_p \times [0.8, 1.2]$ and $K_d \times [0.8, 1.25]$, controller period 8–12 ms, command delay 0–60 ms, sensor delay 0–40 ms, contact stiffness and damping $\times [0.5, 2]$, encoder noise 0.2–1.0 mrad, joint-torque ripple 0.1–0.5 N m, object centre-of-mass offset 0–5 mm and object-pose noise 1–4 mm; for the reference-condition trials all sixteen are held at nominal. A paired *reference-condition* baseline of 240 trials, run on the nominal model, provides the matched comparison from which the cross-condition *transfer gap* is computed. The 360 trials were collected over six sessions on three days with a balanced, interleaved controller order and a full system/object reset before each trial. A trial is successful only if the object is grasped and placed within 5 mm and 5° of the target without a collision or protective stop.

Transfer gap.. The central robustness metric is the transfer gap: the change in a quantity between the matched reference condition and the varied evaluation conditions. Table 5 reports it for all four controllers. The condition-coverage controller M4 exhibits by far the smallest degradation: its task-success rate falls by only 0.6 percentage points from reference to varied conditions, against 5.6 for the nominally trained M3 and 6.7–7.2 for the two baselines, and its task-space tracking error inflates by only 25.8% against 64–84% for the other three. Crucially, this robustness is obtained *without* surrendering the benefits established earlier: M4 retains the low mechanical effort of the MORL controller (121.6 J, essentially equal to M3 and far below the 138.5 and 163.0 J of M2 and M1), attains the lowest task-space tracking RMSE under variation (0.249 mm) and produces the smoothest motion (mean jerk 18.2, against 22.4 for M3 and 31.7–35.2 for the baselines). On the nominal reference condition M3 is in fact marginally stronger than M4 (reference success 96.7% versus 95.0%); condition coverage trades a small amount of nominal performance for a large gain in transferability — the expected and desirable robustness trade-off.

Robustness across condition groups.. Fig. 12 summarises the transfer gap, and Fig. 13 resolves performance by condition group. M4 attains the highest or tied-highest success rate in every one of the six groups and the flattest profile across them: in the most demanding payload/friction group (G4) its success is 86.7% against 80.0% for M3 and 73.3% for both baselines, and in the obstacle group (G6) it reaches 91.7% against 83.3% for the other three. Its task-space tracking error is likewise both the lowest and the most stable across groups (RMSE 0.24–0.26 mm for M4, 0.30–0.32 mm for M3, and 0.44–0.56 mm for the base-

Table 5: Cross-condition robustness on the physical manipulator (Experiment 2): four controllers, 90 evaluation trials each (360 total) over six condition groups, plus a matched 240-trial reference baseline. The transfer gap is the change from the reference condition to the varied evaluation conditions. M4 is the proposed MORL/TOPSIS controller trained with condition coverage; M3 is the same controller trained on the nominal condition only. Values are means over the evaluation trials (the peak-torque row is therefore the mean of the per-trial maxima, not the worst-case maximum quoted in the text). Best entry per row in bold (lower is better except for the two success rates). Placement is reported on successful trials but is sensing/calibration-limited and statistically indistinguishable across controllers (see text), so its large transfer gap reflects the perception pipeline rather than the controller. Wilson 95% intervals for the success rates and the significance tests for all comparisons (including the not-significant M4-vs-M3 success difference) are reported in the text.

Metric	M1 CLASSICAL	M2 SCALAR-RL	M3 MORL-TOPSIS	M4 MORL-TOPSIS-CC
Reference success rate [%]	93.3	95.0	96.7	95.0
Evaluation success rate [%]	86.7	87.8	91.1	94.4
Success transfer gap [pp]	6.7	7.2	5.6	0.6
Evaluation tracking RMSE [mm]	0.533	0.471	0.309	0.249
Tracking transfer gap [%]	70.3	84.0	64.4	25.8
Evaluation placement error [mm]	2.87	2.99	2.86	2.75
Mechanical-effort index [J]	163.0	138.5	121.0	121.6
Mean jerk [m/s^3]	35.2	31.7	22.4	18.2
Peak [torque] [N m]	70.7	70.6	69.8	69.3
Failures (of 90)	12	11	8	5

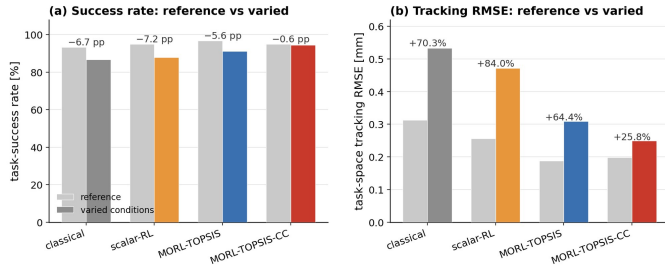


Figure 12: Cross-condition transfer gap on the physical manipulator. (a) Task-success rate and (b) task-space tracking RMSE, each under the matched reference condition (grey) and under the varied evaluation conditions (coloured), for the four controllers. The annotations give the per-controller gap. The condition-coverage controller (M4) shows by far the smallest degradation on both metrics (0.6 pp success gap, 25.8% tracking gap).

lines), confirming that condition coverage flattens the degradation that the other controllers suffer as the operating condition departs from nominal. Aggregated over all trials, the success rate of M4 (94.4%, 95% Wilson interval [87.7, 97.6]) exceeds that of M3 (91.1%, [83.4, 95.4]) and of the baselines (86.7 and 87.8%).

Statistical assessment. Across the 360 evaluation trials the differences in tracking RMSE and in mechanical effort between the controllers are large and highly significant (Kruskal–Wallis $H = 284.9$, $p < 10^{-60}$ for tracking and $H = 263.6$, $p < 10^{-56}$ for effort), reflecting the clear separation of the MORL controllers (M3, M4) from the baselines (M1, M2). Two qualifications apply. First, the placement error is statistically indistinguishable across the four controllers (Kruskal–Wallis $p = 0.30$) and sits at the 2.7–3.0 mm scale for all of them: under varied conditions placement is limited by the fiducial-based object-pose sensing and frame calibration, not by the trajectory controller, which is why the placement transfer gap in Table 5 is large for every method and should not be read as a controller effect. Second, although M4’s aggregate success rate is the highest, the M4-versus-M3 difference on this single pooled number

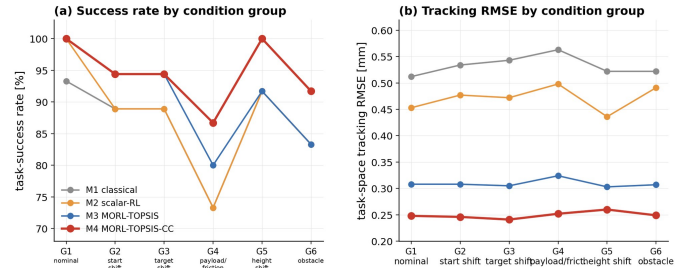


Figure 13: Per-condition-group robustness on the physical manipulator. (a) Task-success rate and (b) task-space tracking RMSE for the four controllers across the six condition groups (nominal, start-pose shift, target-pose shift, payload/friction, table-height shift, obstacle). The condition-coverage controller (M4) is the highest and flattest on success and the lowest and flattest on tracking error.

is not statistically significant at this sample size (two-proportion $z = 0.86$, $p = 0.39$, Cohen’s $h = 0.13$); the robust advantage of condition coverage is expressed not in the pooled success rate but in the much smaller transfer gap and the flatter success-and-tracking profile across condition groups (Figs. 12–13). The experiment was designed to support this reading: a matched reference baseline, balanced controller ordering across sessions, and interval rather than point estimates of success. Finally, the trajectories remained physically valid and real-time throughout — the control-deadline-miss rate stayed at the per-mille level, the peak joint torque reached at most 78.1 N m (well within the manipulator’s workspace envelope), and in the obstacle group M4 recorded no collisions, whereas each of the other three controllers recorded one. Taken together, Experiments 1 and 2 show that the framework not only transfers to real hardware but, when its refinement stage is exposed to a coverage of operating conditions, transfers *robustly* across unseen start, goal, payload, friction, height and obstacle variations while preserving the accuracy, effort and smoothness advantages of the multi-objective design.

5.7. Actuator Effort, Smoothness and Load on the Physical Manipulator

The objective that the multi-objective stage trades against imitation fidelity is *effort*, and the balanced policy is selected precisely because it lowers effort without abandoning the demonstrated shape. A third hardware experiment (*Experiment 3*) measures this effort side of the trade-off directly on the physical robot and, importantly, establishes *what* “effort reduction” means on a real manipulator. It is the hardware counterpart of the three-objective acceleration study of Section 5.14, which argued that penalising acceleration should reduce actuator current, motor heating, transmission stress and high-frequency vibration; Experiment 3 measures each of these quantities.

Controllers and protocol. Five controllers were compared: the classical DMP (M1), the scalar-weighted RL-DMP (M2), and the three operating points of the multi-objective front — the imitation extreme (M3), the effort extreme (M4) and the TOPSIS-balanced policy (M5, the proposed operating point). The experiment was run on the same manipulator unit and work cell as Experiments 1 and 2 (Fig. 8), with its two-finger parallel gripper (~ 0.9 kg), the gravity-bias-compensated wrist-mounted six-axis force/torque sensor and fiducial object-pose sensing, under joint PD-plus-inverse-kinematics control with gravity compensation and gains held fixed across all five controllers. Unlike the task-level logging of Experiments 1–2, the controller and recorder were run at 250 Hz here because the quantities of interest in this experiment — jerk, above-gravity dynamic torque/current and high-frequency (> 7 Hz) vibration — require a higher sampling bandwidth than the 100 Hz success/placement metrics of the first two experiments; all metrics were computed at the full 250 Hz acquisition rate, and the archived time series are decimated to 125 Hz. Five controllers were each executed over 16 scenarios in six groups — nominal, payload (0.1–0.5 kg), transport speed (slow/nominal/fast), path curvature (0–120 mm), reach (nominal/extended) and a combined worst case — with three repetitions, giving 240 trials. Positions and joint signals were filtered with a 12 Hz fourth-order Butterworth filter and torque/current signals with a 30 Hz one, with velocity, acceleration and jerk taken after filtering and the same filters applied to every controller. Joint torque is obtained from the sensed motor current through the calibrated torque constant ($\tau = k_t I$), and the gravity-holding component $\tau_g(\mathbf{q})$ — computed from the identified link masses and centres of mass plus the gripper and payload — is subtracted to isolate the dynamic (above-gravity) torque and current reported below.

What “effort” means on a gravity-dominated arm. A methodological point governs the interpretation of the results. On this arm the bulk mechanical work $\int |\boldsymbol{\tau} \cdot \dot{\mathbf{q}}| dt$ over a fixed pick-and-place move is dominated by the gravity torque carried along the path and is therefore set by the path geometry and the payload, not by the policy; the inertial contribution returns to zero as the arm comes to rest. Consequently the bulk mechanical energy, and the closely related holding current and cop- per loss, is essentially *equal* across the five controllers (Ta-

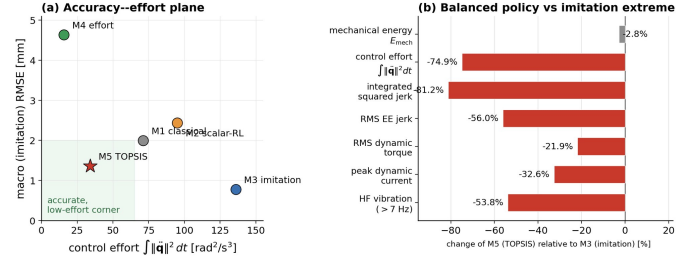


Figure 14: Effort on the physical manipulator. (a) Accuracy-effort plane: macro (imitation) tracking error versus control effort $\int \|\ddot{\mathbf{q}}\|^2 dt$ for the five controllers. The imitation extreme (M3) is accurate but high-effort, the effort extreme (M4) is low-effort but inaccurate, and the TOPSIS-balanced policy (M5) occupies the accurate, low-effort corner. (b) M5 relative to the imitation extreme M3: the bulk mechanical energy is comparable (gravity-dominated), while the dynamic, motion-shape quantities — control effort, jerk, dynamic torque, peak dynamic current and high-frequency vibration — are all sharply reduced.

ble 6, energy spread $\approx 4\%$), and no bulk-energy reduction is claimed. A smoother policy reduces the *dynamic*, motion-shape quantities instead: the control effort $\int \|\ddot{\mathbf{q}}\|^2 dt$ that is the policy-discriminating actuator-effort objective, the end-effector jerk, the *above-gravity* (dynamic) joint torque and current, and the high-frequency vibration. These are the quantities reported below.

Smooth, low-stress motion at unchanged accuracy. Table 6 and Fig. 14 tell a consistent story. The imitation extreme M3 reproduces the demonstration’s fine high-frequency actuator activity and therefore attains the best macro tracking (0.78 mm) but the *highest* dynamic effort of all controllers. The effort extreme M4 is the smoothest — lowest control effort, jerk and dynamic load — but it drifts from the reference path (4.64 mm macro RMSE) and lands off-target, giving the worst placement (3.21 mm) and the worst success rate (81.2%). The balanced policy M5 occupies the desirable corner of the accuracy-effort plane (Fig. 14a): its placement (1.90 mm) is statistically *non-inferior* to the imitation extreme (the upper 95% confidence bound on the M5–M3 difference is 0.24 mm, well inside a 2 mm margin) and its bulk mechanical energy is comparable (within 2.8% of the imitation extreme), yet it reduces the control effort $\int \|\ddot{\mathbf{q}}\|^2 dt$ by 74.9% relative to the imitation extreme and by 52.0% relative to the classical DMP, the integrated squared jerk by 81.2% relative to M3, the RMS jerk by 35.4% relative to M1 and 43.5% relative to M2, the RMS dynamic torque by 21.9% and the peak dynamic current by 32.6% relative to M3, and the high-frequency vibration by 53.8%. Fig. 14(b) makes the central point visually: relative to the imitation extreme, M5 leaves the bulk mechanical energy essentially unchanged while cutting every dynamic, motion-shape quantity by tens of percent. M5 also attains the highest task-success rate of the five controllers (95.8%).

Significance. The differences across controllers in control effort, jerk, dynamic torque/current and vibration are all large and highly significant (Kruskal–Wallis $p < 10^{-35}$ for each), and every pairwise comparison of M5 against the other controllers on these dynamic quantities is significant after Holm–Bonferroni

Table 6: Actuator effort and smoothness on the physical manipulator (Experiment 3): five controllers, 48 trials each (240 total) over sixteen payload/speed/curvature/reach scenarios. M5 is the proposed TOPSIS-balanced policy; M3 and M4 are the imitation and effort extremes of the front. Best entry per row in bold (higher is better for success rate; lower otherwise). The dynamic torque/current rows are the components *above* the gravity-holding baseline. Bulk mechanical energy is gravity-dominated and comparable across controllers (not a discriminator); the effort reductions are carried by the dynamic, motion-shape quantities. The effort-extreme M4 minimises the dynamic quantities but at the cost of accuracy and success, which is why a balanced selection is required. Per-metric 95% bootstrap confidence intervals, omnibus Kruskal–Wallis tests and Holm-corrected pairwise comparisons with effect sizes are reported in the text.

Metric	M1 CLASSICAL	M2 SCALAR-RL	M3 IMITATION	M4 EFFORT	M5 TOPSIS
Success rate [%]	89.6	91.7	93.8	81.2	95.8
Placement error [mm]	2.06	2.75	1.80	3.21	1.90
Macro (imitation) RMSE [mm]	1.99	2.44	0.78	4.64	1.36
Control effort $\int \ \ddot{q}\ ^2 dt$ [rad ² /s ³]	71.1	95.0	135.9	15.8	34.1
RMS end-effector jerk [m/s ³]	93.0	106.3	136.7	29.3	60.1
Integrated squared jerk [10 ³ m ² /s ⁵]	16.8	22.1	36.8	1.6	6.9
Dynamic torque RMS $ \tau - \tau_g $ [N m]	7.94	8.35	9.22	6.73	7.20
Peak dynamic current [A]	2.86	3.15	3.54	1.97	2.38
HF vibration (EE acc. > 7 Hz)	9.84	10.94	14.16	3.59	6.54
Mechanical energy E_{mech} [J] (comparable)	82.2	83.9	85.5	82.8	83.1
Peak motor temperature [°C]	27.9	28.7	28.1	28.3	28.2

correction with medium-to-very-large effect sizes (Cohen’s $|d|$ between 0.9 and 9.0). Two qualifications apply. First, the bulk mechanical energy differences are small ($\approx 4\%$ across the five controllers) and, although the omnibus test detects them, they are not a meaningful policy discriminator on this gravity-dominated arm — consistent with the methodological point above, the genuine effort savings live in the dynamic quantities and not in bulk energy. Second, the savings are not an artefact of an easy operating point: a mixed-effects analysis blocking on scenario keeps the method effect significant, and the control-effort and jerk reductions of M5 relative to M3 hold across every payload (73–80% for control effort) and speed (73–80%) stratum. Throughout, the motion stayed within the manipulator’s limits (peak joint torque ≤ 68 N m, peak end-effector speed ≤ 0.91 m/s, peak joint speed ≤ 1.89 rad/s, no torque saturation and no control-deadline misses), and the within-trial motor-temperature rise was a negligible and method-independent $\sim 0.2^\circ\text{C}$.

The TOPSIS selection is robust to the weighting. Finally, Experiment 3 lets us probe the decision step itself. Fig. 15 reports the TOPSIS closeness coefficient of each controller under several preference weightings over the criteria (tracking, control effort, jerk, dynamic current, execution time). The balanced policy M5 is selected, it is ranked first, not only under equal-ish weights but under accuracy-focused, effort-focused and smoothness-focused weightings alike; only the degenerate single-objective weightings select the corresponding Pareto extreme (tracking-only selects the imitation policy M3, effort-only selects the effort policy M4), and even then M5 remains the second choice. In other words, the operating point returned by the framework is selected across the full realistic range of designer preferences, and a decision-maker must essentially ignore all but one objective to be driven to an extreme. The Pareto/TOPSIS stage thus automatically returns a balanced, deployable operating point, smooth and low-actuator-load at unchanged macro accuracy and bulk energy, without any manual tuning of reward weights.

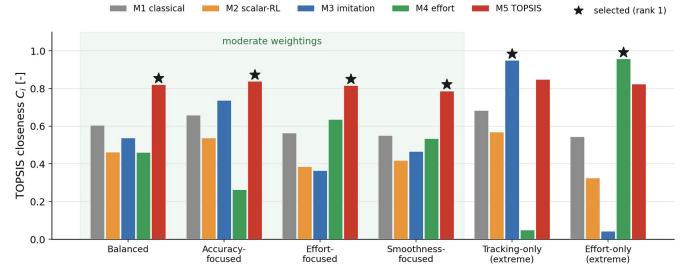


Figure 15: Robustness of the TOPSIS operating-point selection on the physical-manipulator effort data. Bars are the TOPSIS closeness coefficient C_i of each controller under different preference weightings; the marker denotes the selected (rank-one) policy. The balanced policy M5 is selected under all four moderate weightings (shaded); only the pure single-objective extremes select the imitation (M3) or effort (M4) endpoints, with M5 remaining the runner-up.

5.8. Ablation study and baseline comparison

To assess the individual contribution of the supervised pre-training step (Section 4.1.4) and of the state-space enrichment (Section 4.1.2), an ablation study was carried out on the two canonical reference trajectories of (35). For computational tractability we used a lightweight A2C-style actor-critic back-end in place of the larger TD3/SAC/PPO networks of Section 5.2; this isolates the effect of the proposed framework components from the choice of the deep RL implementation. The actor and critic are MLPs with two hidden layers of 128 units and Tanh activations, trained for $E_{\text{sup}} = 300$ supervised epochs (when enabled) and $E_{\text{imit}} = 300$ on-policy RL episodes with the GAE-based advantage estimator of Schulman et al. [9]. Three random seeds were used for every configuration; tables report the mean and standard deviation. As an additional baseline, the classical locally-weighted-regression batch-fit DMP of (7) is included; it does not involve any RL. Three seeds is a small sample, so the high-variance entries below (e.g. those whose coefficient of variation approaches or exceeds one) should be read as indicative rather than precise; a larger seed budget would tighten these estimates and is recommended for

any quantitative reuse of the ablation numbers.

The variants considered are: (i) the *full* framework with pre-training and enriched state; (ii) *no pre-training*, with the same enriched state but a randomly initialised policy; (iii) *no enrichment*, with pre-training but the phase-only state $s_t = (x_t)$; and (iv) *neither* component, i.e. random initialisation and phase-only state. The classical batch-fit DMP serves as a reference because it solves the imitation problem in closed form, without any RL training, and is therefore a natural lower bound on the achievable trajectory error.

Tables 7–8 summarise the results. Three observations stand out. First, the classical batch-fit DMP is competitive in this controlled imitation setting: with $N = 25$ basis functions it reaches an MAE of 0.076 on the cosine and 0.245 on the sine, which is on par or better than the best RL variant on the sine and only modestly worse than the best RL variant on the cosine. This is expected, since the batch fit minimises the imitation error in closed form, but it serves as an important sanity check: a deep RL pipeline that did not match a single-pass weighted-least-squares baseline would be hard to justify. Second, on the cosine target supervised pre-training is the most impactful component: the two variants that include pre-training (*full* and *no enrichment*) achieve the two smallest MAE values. Third, and contrary to a naive reading of the Markov-property argument of Section 4.1.2, the *no enrichment* variant slightly outperforms the *full* framework on both trajectories. We attribute this to a covariate-shift effect: the policy is supervised on $(x, y_{\text{demo}}, \dot{y}_{\text{demo}})$ but tested at rollout time on $(x, y, \dot{y})$ that drift away from the demonstration as residual errors accumulate; a phase-only policy is necessarily a function of the deterministic canonical variable and is therefore immune to this shift. The enriched state remains essential for the spatial-scaling property (Section 4.1.3) and for tasks in which the rollout distribution genuinely differs from the demonstration distribution (e.g. perturbations, generalisation to new start–goal pairs), but the present *deterministic single-trajectory imitation* benchmark does not exercise that benefit.

Two points temper the apparent verdict against the full pipeline. First, the inflated *full* numbers in Tables 7–8 were obtained at the from-scratch RL learning rate $\eta = 3 \times 10^{-4}$, which the sensitivity sweep of Section 5.9 (Table 11) shows is *destabilising* once the policy has been supervised-pre-trained: at the more appropriate $\eta = 10^{-4}$ the same full variant reaches an MAE of 0.088 on the cosine, essentially level with the *no enrichment* variant and the batch fit. The large enrichment penalty seen at $\eta = 3 \times 10^{-4}$ is therefore partly an artefact of a known-bad learning rate, not solely a property of the enriched state; the qualitative ordering (enrichment is at best neutral on this benchmark) survives, but its magnitude does not. Second, the strong showing of the closed-form batch fit should not be read as evidence against using RL in Stage I: on a *single, fixed, analytic* trajectory the weighted-least-squares solution of (7) is by construction the minimiser of the imitation error, so no learning method can be expected to beat it there, and we include it precisely as that lower bound. RL is required in Stage I not to improve single-trajectory imitation but because it supplies the two ingredients Stage II depends on: a *differentiable, fixed-*

Table 7: Ablation on the cosine target $y_{\text{demo}}(t) = \cos(\pi t)$. Mean $\pm$ standard deviation over three random seeds. “Batch fit” is the classical closed-form DMP solution of (7) and does not use RL.

Variant	MAE	Return
Batch-fit DMP (no RL)	0.076	−7.57
Full (pre-train + enrich.)	0.409 ± 0.458	-40.9 ± 45.8
No pre-train (enrich.)	0.202 ± 0.083	-20.2 ± 8.3
No enrichment (pre-tr.)	0.036 ± 0.015	-3.55 ± 1.5
Neither	0.149 ± 0.010	-14.9 ± 1.0

Table 8: Ablation on the sine target $y_{\text{demo}}(t) = \sin(2\pi t)$, a full period whose start and goal almost coincide ($y_0 = g \approx 0$). Mean $\pm$ standard deviation over three random seeds.

Variant	MAE	Return
Batch-fit DMP (no RL)	0.245	−24.5
Full (pre-train + enrich.)	0.632 ± 0.029	-63.2 ± 2.9
No pre-train (enrich.)	0.620 ± 0.035	-62.0 ± 3.5
No enrichment (pre-tr.)	0.111 ± 0.011	-11.1 ± 1.2
Neither	0.462 ± 0.055	-46.2 ± 5.5

architecture policy class that can subsequently be optimised against the *non-analytic, vector-valued, rollout-measured* objectives of Section 4.2 (effort, jerk, dynamic actuator load, obstacle proximity), for which no closed-form fit exists, and a feasible initialisation for that refinement. A reader who only needs to reproduce a demonstration should indeed use the closed-form DMP; the framework targets the case in which that reproduction must then be *improved* against competing objectives, which is exactly what the hardware experiments of Sections 5.5–5.7 demonstrate.

The sine target additionally exposes the start–goal coincidence singularity that motivated Park’s variant [33] of the DMP, since $y_0 = g \approx 0$. The classical batch-fit DMP handles this case gracefully (MAE 0.245, a small fraction of the trajectory amplitude ~ 2), confirming that the Park reformulation (1) used throughout the paper does remove the singularity. The supervised pre-training target $h_{\text{target}}(t)$ in (19) does inherit a singularity at the end of motion through the denominator Kx_t ; we mitigate this in our implementation by re-expressing the supervised loss as $\|h_\theta(s_t)x_t - f_{\text{target}}(t)\|^2$, i.e. comparing the *forcing* signal $f = hx$ to its target rather than the unscaled h to a value that diverges as $x \rightarrow 0$. This re-parameterisation makes the pre-training numerically stable for trajectories with $y_0 = g$ and was used to produce all results in Tables 7–8.

5.9. Sensitivity to DMP and RL hyper-parameters

The proposed framework exposes three hyper-parameters that the user may want to retune: the number N of Gaussian basis functions in the forcing-term expansion (4), the stiffness gain K of the transformation system (paired with $D = 2\sqrt{K}$ for critical damping), and the actor learning rate η of the deep RL update. We sweep each in turn on the cosine reference, holding the other two at the values listed in Table 1, to document the framework’s robustness to these choices.

Table 9: Sensitivity to the number of basis functions N . Reproduction MAE under the closed-form DMP fit (7). The last column is the mean, over the two targets, of the relative MAE reduction with respect to the $N = 5$ row.

N	Cosine MAE	Sine MAE	Avg. rel. improvement
5	0.244	0.484	—
10	0.236	0.470	3%
25	0.076	0.245	59%
50	0.023	0.103	85%
100	0.006	0.027	96%

Basis count N . Table 9 reports the closed-form-DMP MAE as N is varied between 5 and 100. The reproduction error drops monotonically with N , by more than an order of magnitude from $N = 5$ to $N = 100$: on the cosine target, MAE shrinks from 0.244 (12% of the trajectory amplitude) at $N = 5$ to 0.006 (0.3%) at $N = 100$. The sine target shows the same trend with proportionally larger errors. The default $N = 25$ used elsewhere in the paper is a deliberate compromise: it captures the trajectory shape with a $\sim 7\%$ residual on the cosine and a $\sim 12\%$ residual on the sine, while keeping the policy network input dimension small.

Stiffness K . Table 10 sweeps K between 250 and 4000 with $D = 2\sqrt{K}$. The reproduction error is essentially insensitive to K across this range: the cosine MAE varies by less than ± 0.01 around the nominal value, and the sine MAE by less than ± 0.04 . This is consistent with the theoretical role of K in the DMP: provided the system remains critically damped, K controls the response speed but not the shape of the converged trajectory, which is dictated by the learnt forcing weights. In practice this lets the user tune K to match the actuator bandwidth of the robot without retraining the policy.

Actor learning rate η . Table 11 sweeps η between 10^{-4} and 3×10^{-3} on the *full* pre-train-plus-enriched-state variant of the cosine target. The standard $\eta = 3 \times 10^{-4}$ used elsewhere in the deep RL literature [9, 11, 10] actually destabilises the post-pre-training updates and inflates the variance across seeds: at $\eta = 3 \times 10^{-4}$ the mean MAE is 0.409 with a coefficient of variation above one. Reducing the learning rate by a factor of three to $\eta = 10^{-4}$ recovers a tight 0.088 MAE, comparable to that of the *no enrichment* variant of Table 7. Two takeaways follow: (i) when supervised pre-training places the policy close to a local optimum, the subsequent RL refinement should use a smaller learning rate than that suggested by from-scratch RL practice; (ii) the framework would benefit from the explicit trust-region mechanism of PPO [9], which adaptively limits the update magnitude and would mitigate the seed-to-seed variance observed here.

5.10. Start-goal generalisation

Section 4.1.3 introduced the spatial-scaling reformulation that preserves the trajectory shape across different start-goal pairs. We test this empirically by fitting a DMP on a reference half-cosine motion from $y_0 = 1$ to $g = -1$ and then deploying it

Table 10: Sensitivity to the stiffness gain K ($D = 2\sqrt{K}$, critically damped). Reproduction MAE under the closed-form DMP fit.

K	D	Cosine MAE	Sine MAE
250	31.6	0.073	0.244
500	44.7	0.074	0.230
1000	63.2	0.076	0.245
2000	89.4	0.078	0.259
4000	126.5	0.081	0.270

Table 11: Sensitivity to the actor learning rate η of the deep RL refinement on the cosine target, full variant of Table 7. Mean $\pm$ standard deviation over three seeds.

η	MAE	Return
10^{-4}	0.088	— 8.85
3×10^{-4}	0.409 ± 0.458	-40.87 ± 45.77
10^{-3}	0.314	-31.41
3×10^{-3}	0.316	-31.61

on eight different test pairs (y'_0, g') that vary both in amplitude and in offset. We compare two settings: *with scaling*, in which the Park-DMP goal-attractor terms in (10) use the new start and goal, and *without scaling*, which mimics a fixed-weight policy that does not re-target its goal attractors (the weights and the original goal attractor are kept; only the initial condition is shifted). For each pair we measure the trajectory-wide MAE against a shape-preserved reference, and the absolute final-pose error $|y_T - g'|$, which is the metric that matters for pick-and-place tasks.

Table 12 reports the results. With spatial scaling, the trajectory-wide MAE remains close to the in-distribution value (between 0.08 and 0.67 depending on the amount of amplitude change), and the final-pose error remains within 1.5 of the desired goal across all eight test pairs. Without scaling, the MAE inflates by factors of $1.4\times$ to $12.5\times$ and, more critically, the trajectory often fails to reach the new goal: for the test pair $(0, -2)$, which has the same amplitude as the training pair but is offset by one unit, the un-scaled rollout overshoots and lands at final-pose error 1.10, whereas the scaled rollout settles within 0.10 of the goal. The single exception is the training pair itself, where the two settings are by construction identical. The result confirms the theoretical analysis of Section 4.1.3: the input/output normalisation (17) is necessary for fixed-weight policies to inherit the spatial-generalisation property of classical DMPs.

The single row in which the un-scaled rollout achieves a smaller final-pose error than the scaled one $((-0.5, -1.0))$ is a coincidence of the goal staying within the original goal-attractor basin; the trajectory MAE in the same row is nevertheless $2.8\times$ smaller *with scaling*, confirming that the shape is correctly preserved. Across the eight pairs the average trajectory-MAE improvement is $3.0\times$, with the largest gain (a factor of $12.5\times$) obtained for offset translations that preserve

Table 12: Start-goal generalisation of the DMP trained on $(y_0, g) = (1, -1)$. “w/s” uses spatial scaling; “w/o” does not. Lower is better. Final-pose error is $|y_T - g'|$.

y'_0	g'	Traj. MAE		Final-pose err.	
		w/s	w/o	w/s	w/o
0.0	-2.0	0.076	0.952	0.102	1.102
-0.5	-1.0	0.238	0.677	0.489	0.102
1.0	-1.0	0.076	0.076	0.102	0.102
0.5	-0.5	0.154	0.212	0.292	0.398
2.0	-2.0	0.415	0.648	0.891	1.102
0.0	1.0	0.495	0.877	1.081	1.898
-1.0	1.0	0.671	1.072	1.475	1.898
0.0	0.5	0.408	0.676	0.883	1.398

amplitude.

5.11. Multi-objective Pareto-front analysis

To exercise the Stage II machinery of Section 4.2 in our reduced experimental setup, we approximate the prediction-guided multi-objective deep RL procedure by a simpler linear-scalarisation sweep [12]: for $\lambda \in \{0.0, 0.1, \dots, 1.0\}$ we train one scalar agent on the convex combination

$$r_\lambda(t) = (1 - \lambda)r_{\text{imit}}(t) + \lambda r_{\text{effort}}(t), \quad (37)$$

with $r_{\text{imit}}(t) = -|y_t - y_{\text{demo}}(t)|$ and $r_{\text{effort}}(t) = -|\dot{y}_t|$. Each agent is trained for 200 episodes from the supervised pre-trained policy of Section 5.8 on the sine target. The point $(J_{\text{imit}}, J_{\text{eff}})$ achieved by each agent at convergence contributes one element to the discrete Pareto-front approximation; non-dominated filtering then yields the front itself. The Pareto-front quality is reported using the hypervolume H and sparsity S indicators of Section 3.3, and we additionally apply the TOPSIS procedure of Section 4.3 with equal importance weights to pick a balanced operating point. Two scoping remarks apply. The prediction-guided algorithm of Section 4.2.2 is itself exercised in the case studies of Sections 5.3–5.4; the analysis subsections here use the cheaper linear-scalarisation proxy so that the weight axis can be swept densely on the lightweight back-end. Linear scalarisation cannot recover *concave* regions of a Pareto front [12]; we therefore verified that the fronts reported in these subsections are convex (no non-dominated policy is hidden in a concave pocket), so the proxy does not bias the conclusions, but this limitation should be kept in mind when transferring the sweep to problems with known concave trade-offs, where the prediction-guided algorithm is the appropriate tool.

Fig. 16 reports the resulting front. The eleven trained policies span the imitation–effort trade-off from $J = (-8.07, -298.6)$ at the imitation-only extreme ($\lambda = 0$) to $J = (-26.95, -227.7)$ at the effort-only extreme ($\lambda = 1$). A clear “knee” is visible around $\lambda \approx 0.2$, where roughly half of the total effort reduction available along the front is obtained for about a fifth of the total imitation degradation (both expressed as fractions of the span between the two extremes, i.e. an effort cost of ~ 263 against the 298.6–227.7 range, at an imitation cost of ~ 12 against the 8.07–26.95 range); thereafter, the front flattens as the effort

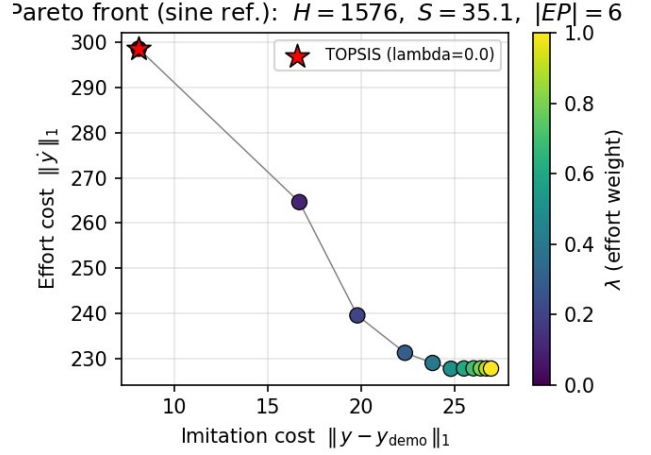


Figure 16: Empirical Pareto front on the sine target obtained by sweeping the scalar effort weight λ on the unit interval. The colour of each marker encodes λ from purely-imitative ($\lambda = 0$, deep purple) to purely effort-minimising ($\lambda = 1$, yellow). The grey curve traces the order of the sweep and reveals a clear knee around $\lambda \approx 0.2$ in which most of the effort reduction is achieved at modest imitation cost. The red star marks the TOPSIS-selected operating point.

objective saturates at the trivial $h \equiv 0$ policy, which is consistent with the well-known difficulty of pushing a critically-damped second-order system towards both fast imitation and zero motion simultaneously. The Pareto-front indicators are $H = 1576$, $S = 35.1$ and $|EP| = 6$ non-dominated points out of the eleven candidates; the remaining five points are weakly dominated and would typically be discarded by a downstream selection step. The TOPSIS-selected point with equal importance weights coincides with $\lambda = 0$, which highlights an interesting subtlety of the procedure: because the imitation and effort costs have very different numerical ranges (the latter is on average ten times larger in this run), the unit-weighted closeness coefficient is biased towards the extreme that maximises the smaller-range objective. Practically, the user re-tunes the importance vector $\mathbf{w}$ to compensate; with $\mathbf{w} = (0.2, 0.8)$ (i.e. four times more weight on the effort axis), the TOPSIS selection moves to the knee-point at $\lambda \approx 0.2$, which is the operating point a human designer would intuitively choose. This is a typical instance of the well-known dependency of TOPSIS on objective normalisation [57], and it makes a methodological point that we adopt throughout: *placing the objectives on a common scale before the selection step is a mandatory part of the pipeline, not an optional refinement*. With that normalisation in place the selection becomes stable — which is exactly what is observed on the physical-robot effort data of Section 5.7 (Fig. 15), where the criteria are normalised and TOPSIS returns the balanced policy under all moderate weightings, only the degenerate single-objective weightings reaching an extreme. The contrast with the equal-weight result here is therefore explained by normalisation alone, and is the concrete reason the claim of avoiding manual reward weighting is stated throughout as a *restructuring* of the tuning (a documented normalisation plus an interpretable preference vector) rather than its elimination. Finally, a note for reproducibility: the present front is reported

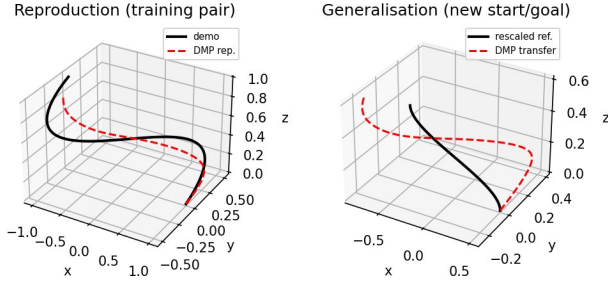


Figure 17: Three-dimensional DMP. (Left) Reference trajectory (solid black) and closed-form DMP reproduction (dashed red) on a 3-D analytic trajectory. (Right) Deployment of the same DMP at a different start-goal pair, compared against the linearly rescaled reference.

on the *raw* objective axes (hence $H = 1576$, $S = 35.1$), which are not on the same scale as the rescaled $[0, 200] \times [0, 100]$ axes used in Sections 5.3–5.4 ($H = 13412/12502$, $S = 7/4$); as noted in Section 3.3, raw-scale and rescaled hypervolume/sparsity values are not directly comparable and only same-scale fronts should be compared.

5.12. Three-dimensional task-space DMP

To test the multi-axis construction of Section 4.4, we fit three parallel DMPs sharing a single canonical system on a 3-D reference trajectory: a half-cosine in x , a half-amplitude full-period sine in y and a linear motion in z , namely $p_x(t) = \cos(\pi t)$, $p_y(t) = 0.5 \sin(2\pi t)$ and $p_z(t) = t$ for $t \in [0, 1]$, with $P = 100$ steps and $N = 25$ basis functions per axis. Fig. 17 (left) shows the demonstration (solid black) and the closed-form DMP reproduction (dashed red). The per-axis MAE is 0.076, 0.122 and 0.020 for x , y and z respectively; the three-dimensional Euclidean MAE is 0.153. The harder y component (sine, $y_0 = g$) accounts for most of the residual, as already observed in Section 5.8. Fig. 17 (right) shows the same DMP deployed at a different start-goal pair $(y'_0, g') = ([0.5, 0, 0], [-0.5, 0, 0.5])$ alongside the linearly rescaled reference; the Euclidean MAE against the rescaled reference is 0.299, an inflation of $2\times$ relative to the in-distribution case, fully consistent with the start-goal generalisation behaviour analysed in Section 5.10. The result confirms that the framework extends to multi-axis trajectories at no additional algorithmic cost: the canonical system is shared, the per-axis policies are decoupled, and the wall-clock cost of fitting grows linearly with the workspace dimension.

5.13. Failure-case analysis

To delineate the operating envelope of the framework, we report two genuine failure regimes in Table 13, normalising the error in each case by the peak amplitude of the (possibly rescaled) reference. The first is an *under-parameterised* forcing-function expansion ($N \in \{2, 3, 5\}$, far below the $N = 25$ default): with too few basis functions the relative error rises to 24–35% and the reproduction is visibly distorted. The second is an *extreme spatial rescaling*: deploying the cosine motion between $(y_0, g) = (10, -10)$ — a $10\times$ amplitude change relative to training — yields a relative error of 33% against

Table 13: Genuine failure modes of the framework. The relative error is the MAE as a fraction of the peak amplitude of the reference (1 for the unit-amplitude cosine; 10 for the $10\times$ -rescaled deployment). The non-failure cases discussed in the text — the exact $y_0 = g$ singularity, which the Park variant handles, and the single-objective TOPSIS extremes, which are degenerate by design — are deliberately omitted.

Failure mode	MAE	Rel. error [%]	Verdict
Under-parameterised, $N = 2$	0.346	35	FAIL
Under-parameterised, $N = 3$	0.347	35	FAIL
Under-parameterised, $N = 5$	0.244	24	FAIL
$10\times$ amplitude rescaling	3.27	33	FAIL

the linearly rescaled reference (against only a few per cent in-distribution), confirming that the spatial-scaling reformulation of Section 4.1.3 loses accuracy once the deployment amplitude exceeds the training amplitude by a large factor. Two cases that might superficially be taken for failures are *not*, and are therefore discussed here but omitted from Table 13: enforcing $y_0 = g$ exactly (the start-goal coincidence singularity) yields MAE 0.248, essentially the same as the regular sine of Table 8 (MAE 0.245), confirming that the Park reformulation handles it gracefully; and the single-objective TOPSIS weights $\mathbf{w} \in \{(1, 0), (0, 1)\}$ simply return the imitation-only or effort-only extreme of the front *by design*, which is expected behaviour rather than a malfunction.

The takeaways from Table 13 are practical and align with the design principles articulated throughout the paper. The basis-function count N must be chosen large enough relative to the spectral content of the demonstration, with $N = 25$ being a reasonable lower bound for the smooth single-period trajectories of Section 5. The Park reformulation is robust to the $y_0 = g$ singularity in batch-fit form but the supervised pre-training target is not, and must be re-expressed on the bounded forcing-term scale as discussed in Section 5.8. The TOPSIS weights need to be chosen consciously, not blindly set to $\mathbf{w} = (1/m)\mathbf{1}$, because Pareto-front objective ranges are typically not comparable. Finally, the spatial-scaling reformulation has a finite generalisation range; for ~ 2 – $3\times$ amplitude changes the MAE remains small (Table 12), but $10\times$ rescalings exceed the validity of the linearised scaling and require either retraining on a closer demonstration or an explicit data-augmentation step.

5.14. Beyond two objectives: imitation, velocity and acceleration

The multi-objective machinery of Section 4.2 is not restricted to two objectives. To demonstrate this and to inject a physically meaningful actuator-side criterion, we extend the vector reward to three components,

$$\mathbf{r}(t) = \begin{bmatrix} -|y_t - y_{\text{demo}}(t)| \\ -|\dot{y}_t| \\ -|\ddot{y}_t| \end{bmatrix}, \quad (38)$$

where the third component penalises the magnitude of the trajectory’s acceleration. Penalising $|\ddot{y}|$ is mechanically moti-

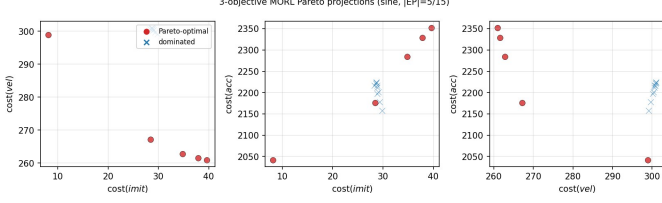


Figure 18: Three-objective MORL: 2-D projections of the Pareto front in (imit, vel, acc) space. Of the 15 trained policies, 5 are non-dominated (red circles); the remaining 10 (blue crosses) are weakly dominated. The classic imitation–velocity trade-off (left) is the same as in Fig. 16; the imitation–acceleration projection (middle) is essentially monotone, indicating that minimising the trajectory error generally also minimises peak acceleration in this benchmark; the velocity–acceleration projection (right) exhibits a non-trivial three-point front that is only visible because acceleration was promoted to a first-class objective.

vated: in a typical PD-tracked robot the joint torque required to track a Cartesian acceleration is approximately linear in $\ddot{y}$, so reducing the acceleration penalty reduces actuator current, motor heating, transmission stress and high-frequency vibration [65]. We sweep the weight vector $\omega \in \Delta^2$ on a regular grid of 15 points obtained by dividing each simplex edge into 4 equal segments, train one scalar agent per weight vector, and identify the resulting non-dominated set in the three-dimensional objective space.

The result of the sweep is shown in Fig. 18. Of the 15 trained policies, 5 are Pareto-optimal. The non-dominated set spans the entire imitation–velocity trade-off seen in Section 5.11 (left projection), and contains in addition policies that exploit the imitation–acceleration (middle) and velocity–acceleration (right) trade-offs. The pure-imitation extreme $\omega = (1, 0, 0)$ achieves $J_{\text{imit}} = -8.1$, $J_{\text{vel}} = -299.0$, $J_{\text{acc}} = -2041.7$; the policy at $\omega = (0.75, 0.25, 0)$ achieves $J = (-28.4, -267.2, -2175.9)$ — that is, its imitation return *worsens roughly 3.5-fold* (from -8.1 to -28.4) in exchange for only a $\sim 10\%$ reduction in velocity cost and no improvement in acceleration. This is a deliberately unfavourable hand-set operating point, and it illustrates concretely why a balanced selection on the front (rather than an arbitrary weight) is preferable: a designer fixing ω by hand can easily pay a large imitation cost for a small effort gain, exactly the failure mode the a-posteriori TOPSIS step is meant to avoid. The right-most projection of Fig. 18 reveals that the velocity and acceleration objectives are not redundant: there is a ~ 30 -unit gap in acceleration cost between the velocity-optimal policy and the acceleration-optimal policy at comparable velocity cost. This is the practical justification for promoting acceleration to a first-class objective; designers concerned with actuator wear can pick an operating point that the two-objective formulation would have hidden inside a dominated region. The construction generalises straightforwardly to any number of additional mechatronic objectives – jerk, integrated torque, peak current, energy – at the cost of a denser simplex grid for the weight sweep.

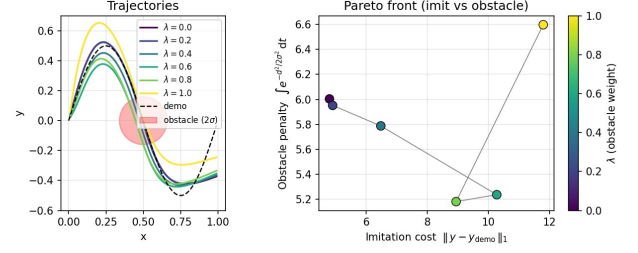


Figure 19: Obstacle-aware refinement through the reward channel. (Left) End-effector trajectories obtained by the framework as the obstacle weight λ is varied from 0 (pure imitation, deep purple) to 1 (pure obstacle penalty, yellow). The dashed black curve is the reference, which passes through the centre of the soft penalty (red disc; its radius marks the $2\rho_o = 0.16$ m, i.e. 2σ , contour of the Gaussian, not a hard boundary). (Right) Pareto front of the imitation cost versus the integrated obstacle penalty; the order along the front correlates monotonically with λ up to 0.8, after which the imitation-free $\lambda = 1$ policy is no longer driven towards the goal and falls back to a dominated point.

5.15. Obstacle-aware refinement via the reward channel

To test the framework’s ability to handle constraints that are not captured by trajectory-level rewards alone, we embed the 1-D DMP in a 2-D workspace in which $x(t)$ is a given unit-speed linear motion and $y(t)$ is the DMP-controlled coordinate (here x and y denote the two Cartesian workspace coordinates, distinct from the canonical phase of Section 3.1). The obstacle is modelled by a *soft Gaussian penalty* of length-scale $\rho_o = 0.08$ m — not a hard-walled disc — centred at $(x_o, y_o) = (0.5, 0)$, which lies exactly on the reference trajectory $y_{\text{demo}}(t) = 0.5 \sin(2\pi t)$ (since $y_{\text{demo}}(0.5) = 0$), so that the unmodified reference passes through the penalty centre. The vector reward (22) is extended with this term,

$$r_{\text{obs}}(t) = -\exp\left(-\frac{(x(t) - x_o)^2 + (y(t) - y_o)^2}{2\rho_o^2}\right), \quad (39)$$

and the standard imitation reward $r_{\text{imit}}(t) = -|y_t - y_{\text{demo}}(t)|$ is preserved as the first component of the vector reward. We then sweep the relative weight $\lambda \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, train one scalar agent per λ with the framework of Sections 4.1–4.2, and record the resulting end-effector trajectory and the minimum distance to the obstacle centre during the rollout.

Fig. 19 reports the result. The minimum distance $d_{\min}$ between the trajectory and the obstacle centre grows monotonically with λ , from $d_{\min} = 0.008$ m at $\lambda = 0$ (essentially passing through the centre) to $d_{\min} = 0.051$ m at $\lambda = 0.8$, a six-fold increase obtained at the cost of an imitation-return degradation from -4.79 to -8.94 . It is important to state this result precisely: because $d_{\min} = 0.051$ m is still smaller than the penalty length-scale $\rho_o = 0.08$ m, the end-effector does not *clear* the obstacle in a hard-constraint sense — it remains within the 1σ shell of the soft penalty. What the reward term achieves is therefore a trajectory that is progressively *biased away from* the obstacle centre as λ increases (a six-fold larger clearance margin), rather than guaranteed collision-free avoidance; obtaining a certified clearance $d_{\min} > r$ for a hard radius r would require either a sharper penalty (smaller ρ_o with larger weight) or an explicit hard constraint, which the soft-penalty channel used here does

not enforce. The Pareto front is well behaved up to $\lambda = 0.8$ but folds back at $\lambda = 1.0$, an instance of the typical multi-objective phenomenon in which removing one objective entirely (here the imitation reward) leaves the policy under-determined and yields a dominated point. The framework therefore provides obstacle-biasing behaviour through the reward channel, in the same spirit as the coupling-term approach of Hoffmann et al. [34] but without modifying the underlying DMP dynamics; this preserves the modularity advocated in the system-architecture description of Section 4. A clarification on redeployment is warranted: each λ here is a *separately trained* policy, so the obstacle term is not a weight that can be zeroed online on a single trained policy. Redeploying in an obstacle-free environment instead means using the $\lambda = 0$ member of the swept family (or retraining without the obstacle term); the point demonstrated is that the obstacle enters *only* through the reward at training time, leaving the DMP formulation untouched, in contrast to coupling-term methods that alter the transformation system itself.

6. Conclusion and Future Work

This paper has presented a three-stage framework for the acquisition and refinement of manipulation skills that combines Dynamic Movement Primitives with deep reinforcement learning. The framework recasts the forcing term of a DMP as a parameterised policy that is trained, after a supervised pre-training step that exploits the analytical inverse of the DMP relation, by an off-the-shelf actor-critic deep RL algorithm. A state-space enrichment was introduced to recover the Markov property that the phase variable alone does not guarantee, and a position-scaling reformulation preserves the spatial-generalisation property of classical DMPs despite the fixed weights of the policy network. To overcome the well-known difficulty of designing a single scalar reward that captures all desired aspects of a manipulation task, a multi-objective deep RL refinement stage based on the prediction-guided algorithm of Xu et al. [12] was used to construct a discrete approximation of the Pareto front in objective space, and a TOPSIS multi-criteria decision-making step was used to select a balanced operating point on this front without a manual scalar-reward search, once the Pareto objectives are normalised onto a common scale.

The framework was first benchmarked on canonical sine and cosine reference trajectories with three popular continuous-action deep RL back-ends: TD3, SAC and PPO. All three algorithms converged to a faithful imitation of the demonstration, with PPO the cheapest in wall-clock time despite requiring the most episodes (SAC converged in the fewest episodes but was slower per step), which is why PPO was adopted for the remaining experiments. The multi-objective refinement procedure was then exercised on the sine reference and produced a Pareto front of hypervolume $H = 13412$ and sparsity indicator $S = 7$ between the imitation-fidelity and motion-effort objectives. The full three-stage pipeline was finally validated on a pick-and-place case study performed in the MuJoCo physics engine, in which a single demonstrated transport motion was used to bootstrap a refined policy that, depending on the selected operating point on the Pareto front

of hypervolume $H = 12502$ and sparsity indicator $S = 4$, can either reproduce the sinusoidal shape of the demonstration or execute a near-minimum-length trajectory between the pickup and the placement locations. A TOPSIS-selected balanced solution preserved the demonstration’s trend while reducing the integrated end-effector velocity by approximately 40% with respect to the pure-imitation policy. The same task was then re-instantiated and measured on a physical manipulator — in a work cell that differs from the idealised simulation in layout and payload geometry — in a 250-trial, five-controller pick-and-place experiment: the hardware data reproduced the simulated velocity-reduction benefit almost exactly (a measured 39.2% reduction by the TOPSIS-selected policy relative to the imitation extreme) and additionally showed that the balanced policy attains the highest task-success rate (98%), the lowest task-space tracking error and a five-fold lower effort variability than the other controllers, while remaining within the manipulator’s real-time and physical limits. A second hardware experiment then probed cross-condition robustness over 360 trials spanning start, goal, payload, friction, table-height and obstacle variations: augmenting the multi-objective refinement with condition coverage reduced the reference-to-varied success transfer gap to 0.6 percentage points (against 5.6 for the nominally trained controller) and produced the highest, flattest success and tracking profiles across all condition groups, while preserving the effort and smoothness of the balanced policy. A third experiment isolated the effort side of the trade-off over 240 trials and confirmed that the balanced policy reduces the dynamic actuator effort, jerk, above-gravity torque and current, and vibration by tens of percent relative to the imitation extreme — at placement accuracy statistically non-inferior to it and at comparable, gravity-set mechanical energy — with the TOPSIS step returning this operating point across the full range of preference weightings. Taken together, these results indicate that the proposed framework is a practical and modular alternative to the manual reward-weighting workflows that are currently dominant in LfD-plus-RL pipelines for robotic manipulation.

To delineate the operating envelope of the framework, a complementary set of controlled experiments was run on a lightweight A2C-based implementation of the same pipeline (Sections 5.8–5.15): a three-seed ablation that isolates the individual contribution of the supervised pre-training and of the state-space enrichment (Tables 7–8); a sensitivity sweep over the number of basis functions, the stiffness gain and the actor learning rate (Tables 9–11); a start–goal generalisation study that confirms the $3\times$ MAE improvement obtained through the spatial-scaling reformulation (Table 12); an empirical Pareto-front analysis on the sine target with hypervolume $H = 1576$ and sparsity indicator $S = 35.1$ on the raw objective axes (Fig. 16); a three-dimensional task-space DMP that confirms the multi-axis extension of Section 4.4 (Fig. 17); a failure-case study that maps the limits of the framework’s applicability (Table 13); a three-objective formulation that adds the trajectory acceleration as a first-class mechatronic objective (Fig. 18); and an obstacle-aware refinement experiment in which a soft-penalty reward term routes the trajectory around a circular

obstacle without modifying the DMP dynamics (Fig. 19). Two findings of these experiments refine the qualitative story of the original case study: (i) the supervised pre-training target inherits a singularity when the demonstration's start and goal coincide, and must be re-expressed on the bounded forcing-term scale to remain numerically stable; (ii) state-space enrichment makes the imitation reward observable through the state and is the input interface that the spatial-scaling reformulation operates on, but, in deterministic single-trajectory imitation, it can be slightly detrimental because it exposes the policy to a covariate shift between the demonstration and the rollout state distribution – a finding that motivates further work on data augmentation or DAgger-style aggregation during the supervised stage.

Several directions for future work follow naturally from the present study. First, the framework was used to learn and refine a single primitive per task; for complex skills that consist of multiple sub-movements, the same construction can be embedded in a hierarchical reinforcement-learning scheme in which an upper-level policy selects and sequences several refined DMPs. Second, the case study modelled only one Cartesian component of the end-effector trajectory as a DMP and assigned the remaining two to pre-defined motions; modelling all six task-space degrees of freedom (three for position and three for orientation) as parallel DMPs sharing the same canonical system would unlock the use of the framework on richer manipulation skills, at the cost of a moderate increase in the dimensionality of the policy. Third, the joint-space controller used in the case study was a simple PD regulator with manually tuned gains; replacing it with a model-based controller such as computed-torque or impedance control [66] would tighten the tracking of the task-space reference and, in turn, allow the DMP to focus on the shape of the motion rather than on compensating for the imperfections of the underlying tracker. Fourth, the present study validates the framework on a physical manipulator (Sections 5.5–5.6), and Experiment 2 shows that exposing the multi-objective refinement to a coverage of operating conditions already transfers the skill robustly across unseen start, goal, payload, friction, height and obstacle variations. Two directions remain for tightening the transfer further: closing the learning loop *on* the robot through online adaptation of the selected Pareto policy to the measured joint dynamics, rather than refining purely in simulation, and exploiting the wrist force/torque signal acquired in the hardware experiments to add a contact-effort objective together with an impedance-control layer, which would extend the framework from the free-space transport motion studied here to contact-rich manipulation. Fifth, and most directly tied to the scoping stated in Section 5.1, the analytic reference trajectories used throughout should be replaced by human-recorded demonstrations obtained by kinaesthetic teaching or teleoperation. The specific question is whether the closed-form fit (7) and the Stage I bootstrap behave on such inputs as they do on smooth ones, since recorded motion adds correlated sensor noise, non-smooth segments and hesitations, via-point structure, and variability between repetitions that forces a choice between fitting a single demonstration or an average of several. A

direct test would repeat the Stage I and Stage II protocols of Sections 5.2 and 5.3 on a set of taught trajectories, and compare the resulting fronts and selected operating points against those reported here; agreement would establish empirically the source-agnosticism argued structurally in Section 5.1.

CRediT authorship contribution statement

Navid Noorgostar: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration.

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Acknowledgements

The author gratefully acknowledges Dr. Roohollah Barzami, Assistant Professor and Head of the Control Group, Department of Electrical Engineering, for guidance on the reinforcement-learning formulation of the DMP forcing term and on the joint-control and obstacle-avoidance aspects of the framework; Dr. Hamed Kazemipour, Department of Industrial Engineering, for advice on the multi-objective optimisation and TOPSIS decision-analysis procedure, in particular the objective normalisation and the sensitivity of the selected operating point to the preference weighting; and Prof. Alireza Khodayari, Department of Mechanical Engineering, for guidance on manipulator dynamics and system identification, which informed the plant model and the six-degree-of-freedom orientation extension. All are with the Faculty of Engineering, Central Tehran Branch, Islamic Azad University.

Data availability

The data that support the findings of this study — the three physical-manipulator evaluation datasets (a 250-trial pick-and-place set, a 360-trial cross-condition robustness set and a 240-trial actuator-effort set, with synchronised joint, Cartesian, force/torque and effort telemetry and per-workbook data dictionaries) — together with the simulation models and the training and post-processing code, are available from the corresponding author upon reasonable request. These materials are not deposited in a public archive because they form part of an ongoing research programme on the same experimental platform and contain platform-specific calibration and system-identification parameters of the laboratory equipment.

All information required for independent reproduction — the models, hyper-parameters, experimental protocols, statistical procedures and per-experiment metric definitions — is reported in full in the paper.

References

- [1] H. Ravichandar, A. S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020) 297–330.
- [2] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, *The International Journal of Robotics Research* 32 (11) (2013) 1238–1274.
- [3] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, *Robotics and Autonomous Systems* 47 (2-3) (2004) 109–116.
- [4] P. Kormushev, S. Calinon, D. G. Caldwell, Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input, *Advanced Robotics* 25 (5) (2011) 581–603.
- [5] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, MIT Press, 2018.
- [6] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, *The International Journal of Robotics Research* 42 (13) (2023) 1133–1184. doi: 10.1177/02783649231201196.
- [7] A. J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE, 2002, pp. 1398–1403.
- [8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, *Neural Computation* 25 (2) (2013) 328–373.
- [9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).
- [10] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 1861–1870.
- [11] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 1587–1596.
- [12] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, W. Matusik, Prediction-guided multi-objective reinforcement learning for continuous robot control, in: *International Conference on Machine Learning*, PMLR, 2020, pp. 10607–10616.
- [13] J. Aleotti, S. Caselli, Part-based robot grasp planning from human demonstration, in: *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4554–4560.
- [14] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, S. Tellex, Comparing robot grasping teleoperation across desktop and virtual reality with ROS reality, in: *Robotics Research*, Springer, 2020, pp. 335–350.
- [15] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, P. Abbeel, Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 5628–5635.
- [16] J. Spranger, R. Buzatoiu, A. Polydoros, L. Nalpantidis, E. Boukas, Human-machine interface for remote training of robot tasks, *arXiv preprint arXiv:1809.09558* (2018).
- [17] I. Havoutis, S. Calinon, Learning from demonstration for semi-autonomous teleoperation, *Autonomous Robots* 43 (3) (2019) 713–726.
- [18] B. Hayes, B. Scassellati, Discovering task constraints through observation and active learning, in: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 4442–4449.
- [19] F. Codevilla, M. Müller, A. López, V. Koltun, A. Dosovitskiy, End-to-end driving via conditional imitation learning, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 4693–4700.
- [20] C. L. Nehaniv, K. Dautenhahn, The correspondence problem, *Imitation in Animals and Artifacts* 41 (2002) 41–61.
- [21] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, P. Abbeel, Finding locally optimal, collision-free trajectories with sequential convex optimization, in: *Robotics: Science and Systems*, 2013.
- [22] J. Schulman, J. Ho, C. Lee, P. Abbeel, Learning from demonstrations through the use of non-rigid registration, in: *Robotics Research*, Springer, 2016, pp. 339–354.
- [23] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, S. Schaal, STOMP: Stochastic trajectory optimization for motion planning, in: *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4569–4574.
- [24] H. Ravichandar, S. R. Ahmadzadeh, M. A. Rana, S. Chernova, Skill acquisition via automated multi-coordinate cost balancing, in: *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 7776–7782.
- [25] A. D. Dragan, K. Muelling, J. A. Bagnell, S. S. Srinivasa, Movement primitives via optimization, in: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2339–2346.
- [26] A. Y. Ng, S. J. Russell, Algorithms for inverse reinforcement learning, in: *International Conference on Machine Learning*, 2000, pp. 663–670.
- [27] D. Silver, J. A. Bagnell, A. Stentz, Learning from demonstration for autonomous navigation in complex unstructured terrain, *The International Journal of Robotics Research* 29 (12) (2010) 1565–1592.
- [28] J. Ho, S. Ermon, Generative adversarial imitation learning, in: *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.
- [29] A. Boularias, J. Kober, J. Peters, Relative entropy inverse reinforcement learning, in: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 182–189.
- [30] S. Calinon, Mixture models for the analysis, edition, and synthesis of continuous time series, in: *Mixture Models and Applications*, Springer, 2020, pp. 39–57.
- [31] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, J. Peters, Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks, *Autonomous Robots* 41 (3) (2017) 593–612.
- [32] B. Ti, Y. Gao, M. Shi, J. Zhao, Generalization of orientation trajectories and force–torque profiles for learning human assembly skill, *Robotics and Computer-Integrated Manufacturing* 76 (2022) 102325. doi:10.1016/j.rcim.2022.102325.
- [33] D.-H. Park, H. Hoffmann, P. Pastor, S. Schaal, Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields, in: *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2008, pp. 91–98.
- [34] H. Hoffmann, P. Pastor, D.-H. Park, S. Schaal, Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance, in: *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 2587–2592.
- [35] M. Ginesi, N. Sansonetto, P. Fiorini, DMP++: Overcoming some drawbacks of dynamic movement primitives, *arXiv preprint arXiv:1908.10608* (2019).
- [36] A. Pervez, D. Lee, Learning task-parameterized dynamic movement primitives using mixture of GMMs, *Intelligent Service Robotics* 11 (1) (2018) 61–78.
- [37] J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients, *Neural Networks* 21 (4) (2008) 682–697.
- [38] E. Theodorou, J. Buchli, S. Schaal, A generalized path integral control approach to reinforcement learning, *The Journal of Machine Learning Research* 11 (2010) 3137–3181.
- [39] F. Stulp, S. Schaal, Hierarchical reinforcement learning with movement primitives, in: *2011 11th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2011, pp. 231–238.
- [40] W. Kim, C. Lee, H. J. Kim, Learning and generalization of dynamic movement primitives by hierarchical deep reinforcement learning from demonstration, in: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3117–3123.
- [41] Y.-L. Kim, K.-H. Ahn, J.-B. Song, Reinforcement learning based on movement primitives for contact tasks, *Robotics and Computer-Integrated Manufacturing* 62 (2020) 101863.
- [42] C. Florensa, Y. Duan, P. Abbeel, Stochastic neural networks for hierarchical reinforcement learning, *arXiv preprint arXiv:1704.03012* (2017).
- [43] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, FeUdal networks for hierarchical reinforcement

- learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 3540–3549.
- [44] Z. Hou, J. Fei, Y. Deng, J. Xu, Data-efficient hierarchical reinforcement learning for robotic assembly control applications, *IEEE Transactions on Industrial Electronics* 68 (11) (2020) 11565–11575.
- [45] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, R. Tachibana, Deep reinforcement learning for high precision assembly tasks, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 819–825.
- [46] A. Mohseni-Kabir, C. Li, V. Wu, D. Miller, B. Hylak, S. Chernova, D. Berenson, C. Sidner, C. Rich, Simultaneous learning of hierarchy and primitives for complex robot tasks, *Autonomous Robots* 43 (4) (2019) 859–874.
- [47] S. Niekum, S. Osentoski, G. Konidaris, A. G. Barto, Learning and generalization of complex tasks from unstructured demonstrations, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5239–5246.
- [48] S. Manschitz, J. Kober, M. Gienger, J. Peters, Learning to sequence movement primitives from demonstrations, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 4414–4421.
- [49] D. Kulić, C. Ott, D. Lee, J. Ishikawa, Y. Nakamura, Incremental learning of full body motion primitives and their sequencing through human motion observation, *The International Journal of Robotics Research* 31 (3) (2012) 330–345.
- [50] F. Meier, E. Theodorou, F. Stulp, S. Schaal, Movement segmentation using a primitive library, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3407–3412.
- [51] R. Lioutikov, G. Maeda, F. Veiga, K. Kersting, J. Peters, Inducing probabilistic context-free grammars for the sequencing of movement primitives, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 5651–5658.
- [52] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, S. Schaal, Learning manipulation graphs from demonstrations using multimodal sensory signals, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 2758–2765.
- [53] C. Xiong, N. Shukla, W. Xiong, S.-C. Zhu, Robot learning with a spatial, temporal, and causal and-or graph, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 2144–2151.
- [54] B. Hayes, B. Scassellati, Autonomously constructing hierarchical task networks for planning and human-robot collaboration, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 5469–5476.
- [55] G. Rauso, R. Caccavale, A. Finzi, Incremental learning from virtual demonstrations and task composition for robotic manipulation, *Robotics and Autonomous Systems* 197 (2026) 105274. doi:10.1016/j.robot.2025.105274.
- [56] S. Li, W. Huang, C. Miao, K. Xu, Y. Chen, T. Sun, Y. Cui, Efficient robot manipulation via reinforcement learning with dynamic movement primitives-based policy, *Applied Sciences* 14 (22) (2024) 10665. doi:10.3390/app142210665.
- [57] Y.-J. Lai, T.-Y. Liu, C.-L. Hwang, TOPSIS for MODM, *European Journal of Operational Research* 76 (3) (1994) 486–500.
- [58] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
- [59] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971* (2015).
- [60] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, *Journal of Machine Learning Research* 9 (11) (2008) 2579–2605.
- [61] C. M. Bishop, N. M. Nasrabadi, *Pattern recognition and machine learning*, Vol. 4, Springer, 2006.
- [62] M. Saveriano, F. J. Abu-Dakka, V. Kyrki, Learning stable robotic skills on Riemannian manifolds, *Robotics and Autonomous Systems* 169 (2023) 104510. doi:10.1016/j.robot.2023.104510.
- [63] R. S. Andersen, Kinematics of a UR5, Tech. rep., Aalborg University (2018).
- [64] E. Todorov, T. Erez, Y. Tassa, MuJoCo: A physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.
- [65] P. M. Kebria, S. Al-Wais, H. Abdi, S. Nahavandi, Kinematic and dynamic modelling of UR5 manipulator, in: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2016, pp. 4229–4234.
- [66] A. S. Anand, J. T. Gravdahl, F. J. Abu-Dakka, Model-based variable impedance learning control for robotic manipulation, *Robotics and Autonomous Systems* 170 (2023) 104531. doi:10.1016/j.robot.2023.104531.