Robotics and Autonomous Systems 191 (2025) 105006

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

From data extraction to data-driven dynamic modeling for cobots: A method using multi-objective optimization☆

Diego Navarro-Cabrera [a,*], Juan H. García-Guzmán [a], Nicolás C. Cruz [a], Brayan Valencia-Vidal [a,b], Niceto R. Luque [a,1], Eduardo Ros [a,1]

[a] Research Centre for Information and Communication Technologies, University of Granada, Granada, Spain
[b] Research Group Osiris & Bioaxis, Faculty of Engineering, El Bosque University, Bogotá, Colombia

ARTICLE INFO

Keywords:
Torque control
Dynamic modeling
Genetic algorithms
PD control
Machine learning
Supervised learning

ABSTRACT

Controlling collaborative robots (cobots) is a new and challenging paradigm within the field of robot motion control and safe human–robot interaction (HRI). The safety measures needed for a reliable interaction between the robot and its environment hinder the use of classical position control methods, pushing researchers to explore alternative motor control techniques, with a strong focus on those rooted in machine learning (ML). While reinforcement learning has emerged as the predominant approach for creating intelligent controllers for cobots, supervised learning represents a promising alternative in developing data-driven model-based ML controllers in a faster and safer way. In this work, we study several aspects of the methodology needed to create a dataset for learning the dynamics of a robot. To this aim, we fine-tune several PD controllers across different benchmark trajectories using multi-objective evolutionary algorithms (MOEAs) that take into account controller accuracy, and compliance in terms of low torques in the framework of safe HRI. We delve into various aspects of the data extraction methodology including the selection and calibration of the MOEAs. We also demonstrate the need to tune controllers individually for each trajectory and how the speed of a trajectory influences both the tuning process and the resulting dynamics of the robot. Finally, we create a novel dataset and validate its use by feeding all the extracted dynamic data into an inverse dynamic robot model and integrating it into a feedforward control loop. Our approach significantly outperforms individual standard PD controllers previously tuned, thus illustrating the effectiveness of the proposed methodology.

1. Introduction

Collaborative robotics is an emerging field dedicated to designing and developing robots capable of safe human-machine interaction, i.e., human–robot collaboration [1]. The control of motion in these collaborative systems is a complex problem since it conjugates both active safety measures, such as torque control to minimize joint applied forces, and passive measures, such as incorporating elastic elements to enhance higher levels of compliance in case of an impact with humans or objects in the environment. These safety measures hinder the calculation of the analytical dynamic model of the collaborative robot (also known as cobot), which prevents the use of classical torque-based control algorithms that rely on simplified rigid dynamic models of robots [2]. Furthermore, position-based control proves unsuitable for human–robot interaction (HRI) due to commanded motion, which can carry significant levels of inertia, resulting in high-impact energy levels that pose a risk to human safety [3].

To overcome the reliance on an analytical definition of system dynamics in traditional control theory, machine learning (ML) is being profusely used [4]. ML offers promising control solutions for operating model-free dynamic systems, enabling accurate and safe task performance. Reinforcement learning, with its capability for generalization and data capture through practice, stands out as a prevalent approach in ML for robotics [5–8]. Nevertheless, this learning approach does come with certain drawbacks for real systems, including a lengthy

☆ The code for the presented work is publicly available in the following repository: https://github.com/EduardoRosLab/From-Data-Extraction-to-Data-Driven-Dynamic-Modeling-for-Collaborative-Robots/. This includes everything necessary to fine-tune PD controllers using MOEAs and the training/deployment of a BRNN trained with extracted data.

* Corresponding author.
E-mail addresses: diegonavaca@ugr.es (D. Navarro-Cabrera), jhelg@ugr.es (J.H. García-Guzmán), ncalvocruz@ugr.es (N.C. Cruz), nluque@ugr.es (N.R. Luque), eros@ugr.es (E. Ros).

[1] Both of these authors contributed equally to this work.

https://doi.org/10.1016/j.robot.2025.105006
Received 4 June 2024; Received in revised form 19 February 2025; Accepted 3 April 2025
Available online 25 April 2025

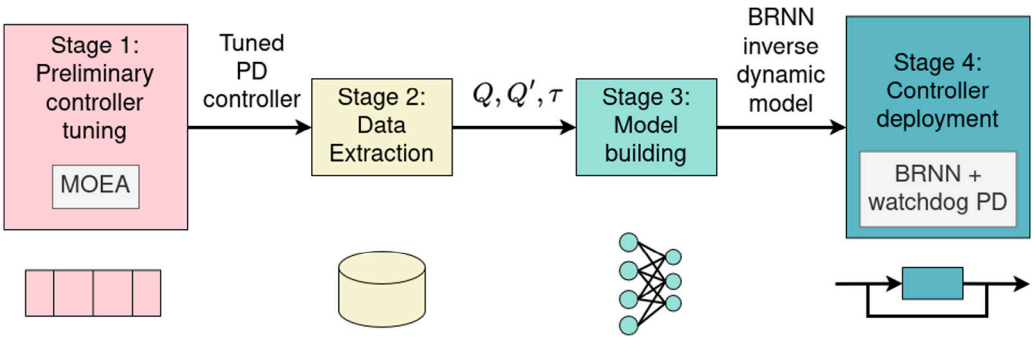


Fig. 1. Stages for implementing and deploying a BRNN-based inverse dynamic model. **Stage 1: Controller Calibration and Tuning** - Define and tune the preliminary PC controller to ensure accurate motion data extraction. **Stage 2: Trajectory Data Acquisition** -Execute multiple trajectories using the tuned controllers to generate a comprehensive dataset. **Stage 3: BRNN Inverse Model Training** - Train the BRNN using the collected position (Q) and velocity (Q') inputs, with torque (τ) as output. **Stage 4: Hybrid Control deployment** -Implement the trained BRNN model in a feedforward + feedback control loop.

learning period and an exploration stage that can pose risks to both the robot and its environment [5,8].

In this work, we focus on developing a methodology to create a dataset that facilitates data-driven learning of a cobot dynamic model, rather than calculating it analytically [9]. Building upon the previous discussion, our main goal is to generate a dataset for studying and developmenting supervised learning models to mitigate breakdown risks during the learning stages with reinforcement learning or other adaptive control alternatives. A data-driven inverse dynamic model takes as input the current position and velocity of each robot joint (denoted as $Q_{current}$, $Q'_{current}$) along with the target state (denoted as Q_{target}, Q'_{target}) and outputs the torque values per joint needed to track the desired motion (denoted as τ). Nevertheless, to capture this input–output relationship accurately, we shall move the robot in a controlled manner, which requires a preliminary controller. Note that the different stages for collecting the data and deploying the data-driven ML controller can be seen in Fig. 1. For this purpose, we use a proportional derivative (PD) controller chosen for its high accuracy when operating at a specific working point and its ease of tuning, as it requires only two parameters per joint [10]. Note that the dataset alone does not directly generalize to new control scenarios without a learned model. The derived data-driven inverse model uses this dataset to generalize and predict appropriate torques for trajectory tracking, adapting to different motion conditions.

An accurate data-driven inverse model shall learn from a wide variety of input–output data. Hence, we require several benchmark trajectories representing different types of motion (to be further explained in Section 2.8). Note that the parameters of the PD controller will depend heavily on the operating points and the specific trajectories used for tuning. Consequently, if we were to use the same PD controller for every trajectory (hereafter denoted as "generic PD controller"), its performance would be inferior compared to using different PD controllers, each fine-tuned for a single trajectory (hereafter denoted as "fine-tuned PD controllers"). We will further illustrate this difference in the Results section. The PD controller is adjusted through evolutionary multi-objective optimization, prioritizing movement precision and torque values to ensure safety (see Fig. 2). Several algorithms are used for comparative purposes, i.e., NSGA-II [11], SPEA2 [12] and HypE [13].

Fine-tuning the PD controller requires precise adjustment of its parameters for each benchmark trajectory. Each data sequence of torque value and reached position, obtained from individual PD adjustments, is generated specifically to train a subsequent ML controller. This ML controller will be able to generalize the control action and adapt it to different types of trajectories [15]. PD control is widely used in robotic manipulators due to its simplicity [10], minimal adjustment required for each robot joint i.e., only two parameters per joint, and accuracy for simple tasks within a limited range of motion. Note that the parameters of the PD controller will depend heavily on the operating points and the specific trajectories used for tuning.

Previous studies have extensively explored the use of MOEAs in fine-tuning PD controllers across diverse robotic applications. For instance, in [16], a Proportional–Integral–Derivative (PID) controller was tuned using NSGA-II. Many of these studies were designed for robotic manipulators. However, they were mostly used in simplistic planar two-degree-of-freedom (d.o.f.) simulated mechanical models [16–21], which do not resemble those commonly used in real-life industrial applications [22]. In contrast, this work investigates diverse aspects of using MOEAs in fine-tune PD controllers, particularly focusing on the more complex KUKA LBR iiwa robot arm equipped with 7 d.o.f. [23].

After fine-tuning a set of PD controllers to different benchmark trajectories, the proposed dataset captures the motion state of the cobot (joint positions and velocities) and the commanded torque values. This allows us to model the relationship between desired motion and torque control. Depending on the direction of this relationship (reached position to applied torque values or vice versa), the dataset can serve to build either an inverse dynamic model or a forward dynamic model of the cobot. Furthermore, using the inverse dynamic model of the system, we can create a ML-based feedforward controller to work conjointly with a PD controller. This ML controller will be able to generalize the control action, adapting to different types of trajectories [15], thereby overcoming one of the major drawbacks of PD control. We will try this approach using a Recurrent Neural Network (RNN) as the ML controller.

Similarly, in [15], we conducted an experiment focusing on identifying the types of trajectories best suited for inverse dynamic ML learning. Building on that work, here, we shift our focus to the quality of the data being learned, beyond just the trajectories themselves. This quality is achieved by using a multi-objective approach for fine-tuning PD controllers. We compare the performance of PD controllers fine-tuned to each specific trajectory in the dataset with more generic all-purpose controllers tuned to handle all trajectories simultaneously.

The main contributions of the presented work are as follows:

- Proposal and application of a methodology for dataset creation in data-driven dynamic modeling: We describe the steps needed to create a dataset fit for ML-based dynamic modeling. This dataset links the state of the robot to the corresponding torque commands per joint, thus enabling the creation of either a data-based inverse or a direct dynamic model using ML techniques.
- Data validation through model construction: We validate our dataset and methodology by training two RNN models to learn the inverse dynamic model of a simulated KUKA iiwa cobot. One model is trained using data from the fine-tuned PD set while the other uses data from the generic PD set. The resulting RNN dynamic models outperformed even the fine-tuned PD torque controllers when used as feedforward controllers in a real-time

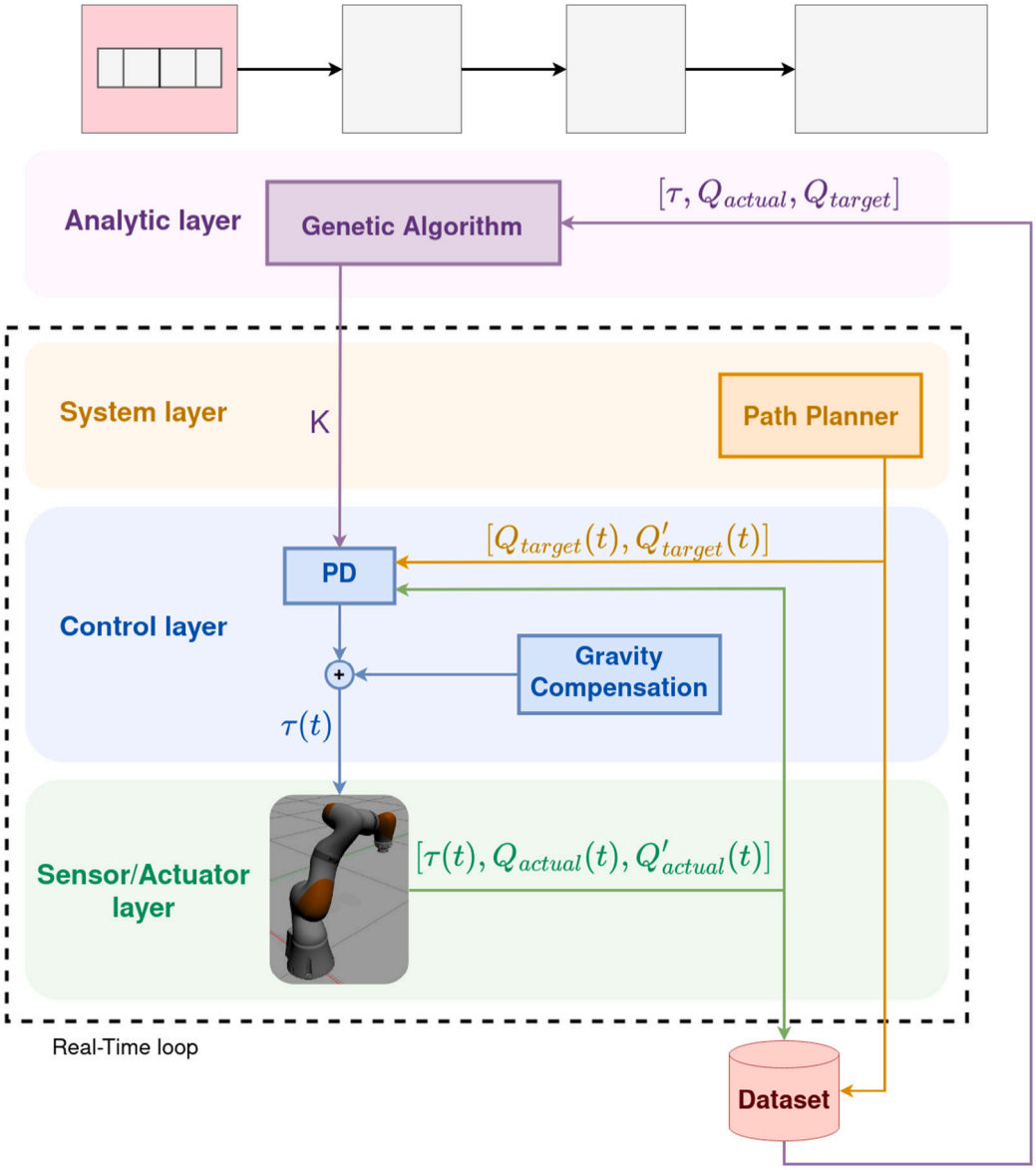


Fig. 2. Control system schematic view: This figure shows the PD torque control loop flow diagram. The 4-layer architecture is adapted from EU project IMOCO4.E [14] to our methodology. The system layer reads the desired trajectory and sends the desired position and velocity at time t ($[Q_{target}(t), Q'_{target}(t)]$) to the controller; the control layer sends torque commands ($\tau(t)$) to the cobot; and the sensor layer returns the current state ($[Q_{actual}(t), Q'_{actual}(t)]$). Extracted data is saved for asynchronous use by the analytic layer to update the PD controller gains (K). $[\tau, Q_{target}, Q_{actual}]$ represent arrays, while $[\tau(t), Q_{target}(t), Q_{actual}(t)]$ represent the tth element of each array.

control loop. This significant performance improvement remarks the importance of high-quality datasets for fitting data-driven dynamic models, used as torque controllers.

The rest of the article is structured as follows: Section 2 describes in detail the methodology for capturing the proposed dataset. Then, Section 3 presents the experimentation and results obtained. Finally, Section 4 shows the conclusions and challenges for future works.

2. Materials and methods

2.1. Multi-objective optimization

Optimization problems arise in multiple fields, from Engineering to Applied Sciences [24,25]. These problems typically seek the extrema of functions representing aspects of interest, which depend on different variables. The function defines whether one seeks minima or maxima, which particularizes the problem into minimization or maximization, respectively. In our context, one can view the parameters of a PD controller as the decision variables, and the resulting accuracy of movement as the performance criterion (to be maximized in our case).

Mathematically, let f be a function in which we are interested in finding its minimum (point and value). It is possible to define the following optimization (minimization) problem:

$$\begin{aligned} &\text{minimize } f(x) \\ &x \text{ subject to } L_i \leq x_i \leq U_i, i = 1, \ldots, n, \end{aligned} \tag{1}$$

where f is an n-dimensional function of the form $f : \mathbb{R}^n \to \mathbb{R}$ called the objective function, and x refers to a generic input in $[L_1, U_1] \times \cdots \times [L_n, U_n] \in \mathbb{R}^n$. Accordingly, feasible inputs belong to a vector subspace in $\mathbb{R}^n$ that is limited by a lower and an upper bound in each dimension i, i.e., L_i and U_i for $i = 1, \ldots, n$, respectively. This space is known as the feasible or search space. Based on this formulation, minimization and maximization are virtually equivalent because finding the minimum of $f(x)$ is the same as obtaining the maximum of $-f(x)$.

Sometimes, there is no further information about the objective function, e.g., an analytical expression directly relating the variables, and the constraints are variable bounds only. Then, the problem can be defined as a black-box optimization problem with box constraints [24,26], which fits well with model-tuning applications where the objective function ultimately depends on a computer simulation [24,26]. It is also the most common application case for evolutionary algorithms and metaheuristics in general, as it is not feasible to obtain the solution analytically [24,25,27]. This is the approach followed in [24,28,29].

However, in real-world problems, it may be necessary to consider several objective functions in conflict [30,31]. Fortunately, the previous formulation can be extended to deal with multi-objective problems with m objective functions, $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ as follows:

$$\begin{aligned} &\text{minimize } F(x) = (f_1(x), \ldots, f_m(x)) \\ &x \text{ subject to } L_i \leq x_i \leq U_i, i = 1, \ldots, n. \end{aligned} \tag{2}$$

In this context, the m-dimensional vectors $F(x) = (f_1(x), \ldots, f_m(x)) \in \mathbb{R}^m$ are known as objective vectors. It is also common to call decision vector to any feasible n-dimensional input of variables, $(x_1, \ldots, x_n)$. Maintaining the single-objective scheme, solving the problem in Eq. (2) would imply finding the decision vector that results in the objective vector with the lowest possible value for each component. However, provided that there are objectives in conflict, i.e., the components of $F(x)$ are competing, the solution to the problem is not unique. Instead, it consists of multiple trade-off points of the feasible space, and the user who launched the optimization process must choose the decision vector to consider [30,31]. For this reason, this person is usually called the decision maker in multi-objective optimization.

This situation can be illustrated within the context of this work; consider two different sets of PD controller parameters for operating a robotic arm. One set yields the highest accuracy, while the other ensures the safest operation in terms of applied forces. As solutions, both are virtually equivalent, and it all depends on the final decision. However, before that point, it has been possible to discard other options as sub-optimal. The reason is that two different objective vectors can be compared according to dominance, which is a key concept in multi-objective optimization. Specifically, provided two objective vectors x and y, x dominates y, expressed as $x \succ y$, if none of the components of x, i.e., $F(x) = (f_1(x), \ldots, f_m(x))$ is worse (higher, assuming minimization) than its equivalent in y, i.e., $F(y) = (f_1(y), \ldots, f_m(y))$, and there is at least one in $F(x)$ that is better than its analogue. Mathematically [30]:

$$x \succ y \leftrightarrow \begin{cases} \forall i \in 1, \ldots, m : f_i(x) \leq f_i(y), \\ \exists j \in 1, \ldots, m : f_j(x) < f_j(y). \end{cases} \tag{3}$$

If an objective vector dominates another, the decision vector (input variables) producing the former is intrinsically better than that of the latter, which can be discarded as a solution. However, neither may dominate the other. In this situation, both vectors are called indifferent to each other. The solutions to the problem in Eq. (2) will be decision vectors generating indifferent non-dominated objective ones. Accordingly, solving a multi-objective problem involves finding the decision vectors linked to non-dominated objective ones. These points are also known as Pareto-optimal or optimal in the Pareto sense. They form the Pareto optimal set, and their associated objective vectors define the Pareto front. Section 2.2 further particularizes these concepts for the problem at hand, and Section 3.5 explains how to select points from the Pareto front.

2.2. PD controller tuning: Optimization problem formulation

For a robot with n joints, a PD controller is defined by a proportional constant (KP_i) and a derivative one (KD_i) for each joint i. Thus, such a controller can be defined by a vector $K \in \mathbb{R}^{2n}$ according to Eq. (4).

$$K = (\mathrm{KP}_0, \mathrm{KD}_0, \ldots, \mathrm{KP}_n, \mathrm{KD}_n). \tag{4}$$

When controlling collaborative robots there are 2 main problems to be addressed. First, accuracy in trajectory tracking; the cobot must be accurate when following a specified trajectory. This is quantified using the mean Euclidean distance, commonly referred to as the Mean Average Error (MAE), between the end-effector and the desired Cartesian coordinates (benchmark trajectory) along the trajectory. See Eq. (5), where $Q(t)_{target}$ and $Q(t)_{actual}$ represent the target and actual coordinates of the end-effector at time t, respectively.

$$f_a(Q_{target}, Q_{actual}) = \frac{1}{T} \sum_{t=1}^{T} |Q(t)_{target} - Q(t)_{actual}|. \tag{5}$$

This metric is commonly used when evaluating the accuracy of robotic manipulators and collaborative robots [32], providing a robust estimation of the robot performance in executing the desired task.

The second problem lies in robot safety for HRI; the robot torque values commanded by the controller must remain low and change smoothly, avoiding sudden acceleration peaks. To measure this, we use Eq. (6), where τ represents the vector of commanded torque values, T denotes the number of steps in a trajectory, and $\tau(t)$ corresponds to the torque applied at time t.

$$f_t(\tau) = \frac{1}{T} \sum_{t=1}^{T} (\tau(t) - \tau(t-1))^2. \tag{6}$$

We use the squared power of the derivative. This approach allows us to measure how much torque is used throughout the trajectory while penalizing any sudden spike in torque that may cause unsafe and unpredictable behavior, similar to the TV metric presented in [33].

In general, higher KP and KD values improve the accuracy of the control system by letting it take faster and more intense reactions to trajectory deviations. However, when they become too high, they can cause overshoots and oscillations. High PD values also lead to higher torque values and potentially abrupt torque changes, which is unsafe for HRI. Furthermore, the kinematic chain constituting the robot makes the joint motions influence each other, i.e., inertia coupling effects [9]. Therefore, the KP and KD parameters of each joint must be tuned simultaneously to find trade-off configurations.

For a given configuration vector, K, the only way to know the MAE and τ in a specific trajectory is by executing it. Consequently, let us define an abstract function *followPath(trajectory, K)* that executes the trajectory using the specified PD parameters (K) and returns the tuple ($Q_{\text{target}}, Q_{\text{actual}}, \tau$). This imaginary function may refer to a real experiment or an accurate simulation, as in this work. In either case, it allows computing Eqs. (5) and (6), which ultimately let us reformulate them as $f_a(K)$ and $f_t(K)$, respectively. When using more than one trajectory for PD tuning, as is the case with generic PD controllers, an average across all trajectories is calculated for f_a and f_t .

With this modification, we can define the target optimization problem as expressed in Eq. (7). Solutions to this problem maximize the accuracy and safety of the target cobot by minimizing the MAE and torque values applied, respectively.

$$\begin{aligned} &\text{minimize } F(K) = (f_a(K), f_t(K)) \\ &K \text{ subject to } 0 < K_i < U_i, i = 1, \ldots, 2n, \end{aligned} \tag{7}$$

where n is the number of joints, and U_i is the upper bound of the ith element in vector K. This upper bound is set to five times the maximum torque value for each joint. This approach ensures that joints with higher precision but lower power have a smaller exploration range than those with more power but less accuracy.

For the sake of computational efficiency, the discrete-time derivative of the error will be calculated as $\frac{de}{dt} = (e_t - e_{t-1})$, without normalizing by the control loop frequency (250 Hz in our case). Consequently, the derivative error will be amplified by a factor of 250. To compensate for this amplification, the maximum values of the derivative components of the PD controller will be divided by 250. Taking all these considerations into account, the maximum values for

each variable will be KP = {1600, 1600, 880, 880, 550, 200, 200}, KD = {6.4, 6.4, 3.52, 3.52, 2.2, 0.8, 0.8}.

Once the optimization process is completed, a Pareto front is generated where each point in the Pareto front represents a specific PD controller configuration derived during the data extraction (Fig. 1, Stage 1 and 2). In this context, each point corresponds to a candidate PD controller with a specific trade-off between accuracy and safety. The resulting Pareto front set the decision-maker to choose a PD controller. Please see Section 3.5 for further details regarding PD controller selection inside the Pareto front.

The evaluation and comparison of the solution sets provided by the MOEAs when adjusting PD gains relies on the Hypervolume Indicator (HV) [34], a well-established metric in multi-objective optimization. It is defined around the concept of dominance between solutions. As explained in Section 2.1, a solution s is said to dominate another solution t when s outperforms t in at least one objective without obtaining inferior results in the rest [35]. Based on this idea, the HV quantifies the quality of the solutions found by a MOEA by measuring the volume of the objective space they dominate (Fig. 4.a). More specifically, given a set of sorted non-dominated solutions S generated by a MOEA, which defines a Pareto set, and a reference point R (typically set to a point dominated by all solutions), the HV is calculated according to Eq. (8).

$$\begin{aligned} \mathrm{HV}(S,R) = &\sum_{i=1}^{N-1}((S_a[i+1]-S_a[i])*(R_t-S_t[i])) \\ &+((S_a[N]-R_a)*(R_t-S_t[N])), \end{aligned} \tag{8}$$

where $S_a[i]$ and $S_t[i]$ refer to the objective values of the ith solution in the Pareto front according to f_a and f_b, respectively. Similarly, R_a and R_t are the objective values of the reference point in the corresponding objectives. Finally, N is the size of the Pareto front.

The HV is typically normalized by the volume of the entire objective space, resulting in HV_{norm} (see Eq. (9)). This way, it provides a scalar value between 0 and 1, where 1 indicates the ideal situation in which the entire objective space is covered. This metric provides a comprehensive comparison criterion of the spread and convergence rate of the Pareto fronts found by the different MOEAs. Accordingly, all HV values used in Section 3 will be normalized.

$$\mathrm{HV}_{norm}(S,R) = \frac{HV(S,R)}{R_a R_t}. \tag{9}$$

Given the wear and tear occurring in the robot during the tuning process, achieving fast convergence is essential. Thus, a strict termination criterion is necessary. In this work, we will use the stability of the evolution of the HV as a termination criterion. A MOEA will be considered to have converged when the HV (rounded to three decimals) stops improving for five consecutive generations, at which point the algorithm halts. A generation is considered not to have improved when the difference between the HV defined by its current population of solutions and that of the previous one is less than 0.001. It is also noteworthy to mention that even if the MOEA continues to find new Pareto-dominant solutions, it automatically halts upon reaching 4000 evaluations. This limit is set to ensure computational efficiency and prevent excessive runtime, as improvements beyond this point tend to be marginal while significantly increasing computational cost.

2.3. MOEA algorithms: Pseudo-code for NSGA-II, SPEA2 and HypE

In the selection of the MOEA for dataset creation, we evaluated the performance of three different algorithms: NSGA-II [11], SPEA2 [12], and HypE [13]. We also compared the results of these three algorithms with those of a random search, serving as a baseline. Additionally, we provide the pseudo-code for each algorithm to offer the reader a clearer understanding of their intricacies and functionality, particularly tailored to our specific use case. The detailed pseudo-code fragments are presented in algorithm 1.

The initial population is generated randomly, with each variable taking values between 0 and an upper limit as discussed in Section 2.2.

2.4. Analytical cobot dynamic modeling and its limitations

The dynamic model of a robot describes the relationship between the torque applied by the joint motors and the resulting motion. This relationship is typically articulated through the Lagrange formulation as expressed below:

$$\tau = M(q)q'' + C(q,q') + g(q) + \varepsilon(q,q',q''), \tag{10}$$

where, (q, q', q'') terms represent the joint positions, velocities and accelerations, respectively. The terms $M(q)$, $C(q,q')$, and $g(q)$ correspond to the inertia matrix, Coriolis effect, and gravitational pull, respectively. Finally, the term $\varepsilon(q,q',q'')$ encompasses factors that are not considered in the dynamic model, such as friction effects or the elastic components within the cobot. In the dynamic models of many rigid robots, the term $\varepsilon(q,q',q'')$ is assumed to be negligible since their high-ratio gear boxes cause M(q) and C(q,q′) dynamic terms to have a rather higher impact than $\varepsilon(q,q',q'')$.

Conversely, in cobotics, safety features, such as less powerful motors in the joints or the presence of elastic components, amplify the influence of $\varepsilon(q,q',q'')$ in the dynamics of the cobot. Consequently, an expansion of the original formulation becomes imperative.

While several works have pursued the modeling of the friction component [36,37], modeling becomes more complicated when dealing with the effects of elastic components [38]. Therefore, the dynamic modeling of collaborative robots remains a challenge, demanding not only an accurate representation of rigid body dynamics, but also capturing the elastic behavior inherent in the cobot joints. Due to the mathematical intractability caused by these elastic behaviors, cobot dynamic modeling is prone to be learned rather than calculated, unlike traditional rigid industrial robots [32,39,40].

2.5. The collaborative robot

The lightweight robot, LBR iiwa, from KUKA® was used as our virtual robotic demonstrator. LBR iiwa is an industrial robot specifically designed for human–robot collaboration (HRC), and as such, it integrates several sensors to measure torque in all of its joints and offers the Fast Research Interface (FRI), able to send torque commands from an external computer. This interface will allow us to use torque control to balance accuracy and safety. The connection between the PD controller and FRI has been done using the interface provided by [41]. All experiments were conducted in simulation, using the Gazebo simulator [42]. The simulation was reset after each trajectory to ensure determinism during controller testing.

2.6. Data acquisition method: Torque control loop

To create a data-driven dynamic model, it is mandatory to capture information about the relationship between the torque and motion. For this purpose, the cobot must execute specific trajectories i.e., benchmark trajectories. However, to allow trajectory execution, a preliminary torque controller is essential. The type of motion generated by this preliminary torque controller serves as the basis for the final dynamic cobot model, and as such, the accuracy and safety of this preliminary torque controller are crucial for obtaining high-quality data. As in [15], where different trajectories are used, each fine-tuned with a specific PD controller, here we propose finding optimal examples of motion that can then be extrapolated into a general-purpose controller using ML. PD controllers, known for their simplicity, are widely used in motion control [10]. As demonstrated later in this study, while not suitable for general applications, they exhibit remarkable accuracy when precisely tuned to a specific motion (see Section 3.4). In the creation of our dataset, we will use a torque-based PD controller, with gravity compensation provided by the LBR iiwa internal controller. The torque control loop is illustrated in Fig. 2.

Algorithm 1 MOEA for PD tuning. 3 different algorithms are highlighted in color: NSGA-II SPEA2 HypE

Require: Population of PD controllers: C, Trajectory: T, Archive of Non-dominated Controllers: NC
$NC = \emptyset$
$NC = \emptyset$
$\forall c \in C$: initialize c.KP and c.KD randomly
$\forall c \in C$: c.f_a, c.f_t = evaluate(T,c)
$\forall c \in C$: assign c.fitness based on Pareto dominance and crowding distance
Binary Tournament Selection in C
Add non-dominated individuals to NC
Binary Tournament Selection in NC
Add non-dominated individuals to NC
Binary Tournament Selection in NC
SBX crossover and Polynomial mutation → c
while Hypervolume Indicator not stable **do**
    c.f_a, c.f_t = evaluate(T,c)
    Assign c.fitness based on Pareto dominance and crowding distance
    if $\exists x \in C \rightarrow c.fitness > x.fitness$ **then**
        Add c to C
        Remove x from C
    end if
    Binary Tournament Selection in C
    if c is not dominated in $C \cup NC$ **then**
        Add c to population
        if NC capacity is exceeded **then**
            Remove $x \in NC$ based on neighbor euclidean distance
        end if
    end if
    Binary Tournament Selection in NC
    if c is not dominated in $C \cup NC$ **then**
        Add c to population
        if NC capacity is exceeded **then**
            Remove $x \in NC$ with a lesser contribution to Hypervolume
        end if
    end if
    Binary Tournament Selection in NC
    SBX crossover and Polynomial mutation → c
end while

2.7. Data acquisition method: 4-layer architecture

The proposed method for data gathering follows a 4-layer architecture (Fig. 2) inspired by [14,43]. This 4-layer architecture will be interconnected using ROS2 (Robot Operating System 2) [44] and can be described as follows:

- **Sensor/Actuator Layer:** This layer comprises sensors and actuators used by the LBR iiwa cobot. It receives instructions from the controller and provides data on joint states.
- **Control Layer:** The PD controller is located at this layer and receives information regarding the next desired set-point. It uses the PD parameters (proportional gain, KP, and derivative gain, KD) and sends the corresponding torque commands to the motor actuators.
- **System Layer:** Responsible for path planning, this layer sends data related to the desired benchmark trajectory to be followed to the Control Layer.
- **Analytic Layer:** In this layer, MOEAs are used. These algorithms adjust PD controller gains based on accuracy and safety measurements integrated into their fitness functions, ensuring precision and safety in the resulting motion.

The System, Control, and Actuator/Sensor layers operate on a real-time loop at 250 Hz frequency. Within this time frame, the System layer sends the benchmark trajectory to the Control layer, which, in turn, transmits the torque commands to the cobot and receives the updated sensor data. The torque value, position, and velocity of each joint are recorded in an array and written to a file once the trajectory is completed. Subsequently, the analytic layer reads this data file to evaluate performance and communicates asynchronously with the Control layer to update the PD gains. This division ultimately facilitates the scalability of our methodology by separating the Analytic and System layers from the Control and Sensor ones. Furthermore, it enables the use of multiple cobots following the same trajectory in parallel.

2.8. Benchmark trajectories for dataset inclusion

Incorporating benchmark trajectories into our dataset aligns with the findings from [15]. Specifically, we incorporate spiral and random trajectories, chosen for their ability to generate meaningful datasets without unnecessary data volume [15]. This selection ensures that the dataset remains suitable for the efficient training of ML controllers. We also use rectilinear trajectories, which combine linear movements

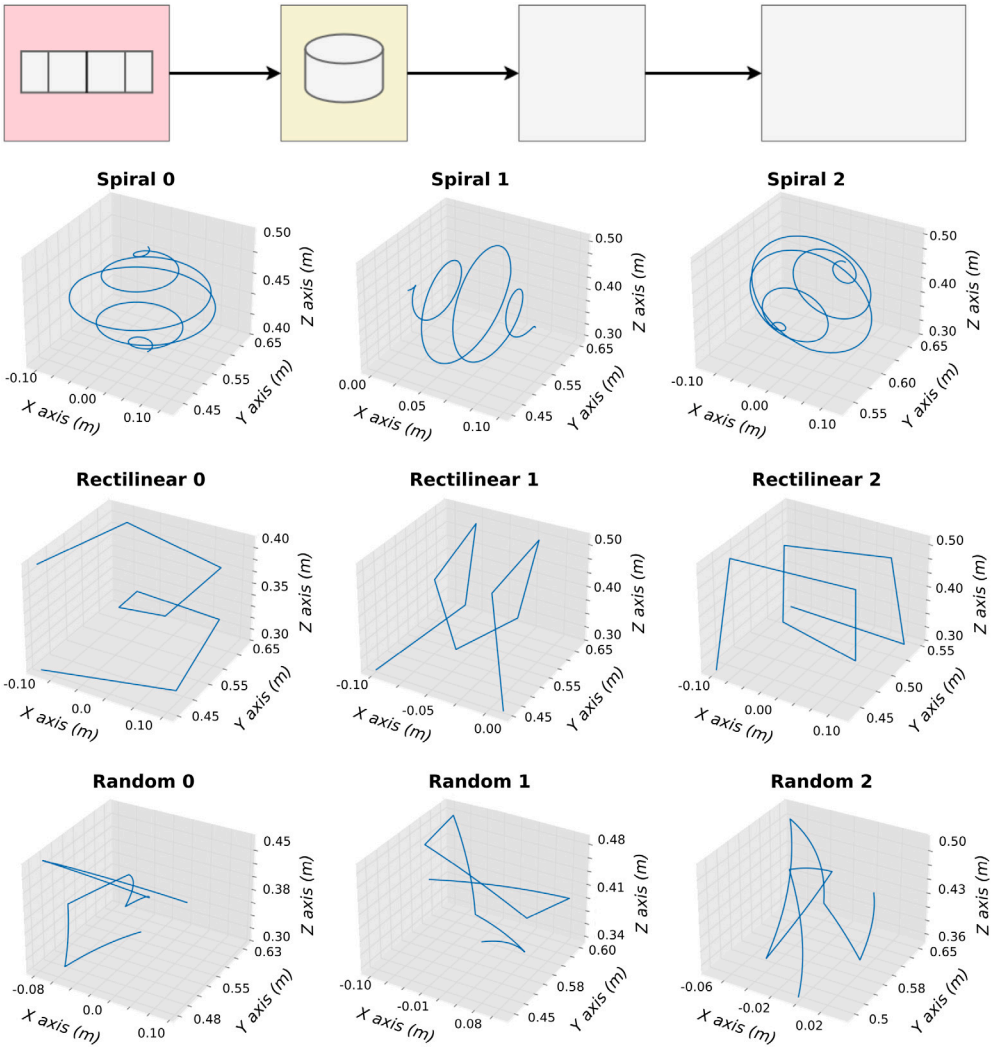


Fig. 3. Benchmark trajectories used for data gathering. Snapshots of the different benchmark trajectories included in the dataset: spiral, rectilinear, and random trajectories.

with sharp turns. These trajectories aim to enhance the ML controller understanding when operating under high acceleration and velocity gradients, thus resulting in larger inertia values. Examples of the trajectory dataset are illustrated in Fig. 3. It is necessary to differentiate between traditional dynamic model identification, which relies on excitation trajectories for analytical modeling [45], primarily for rigid-bodied robots, and data-driven dynamic models, which use motion examples to learn optimal torque commands for desired movements, especially for non-rigid-bodied cobots [15]. Consequently, trajectories suitable for one form of modeling may not be suitable for the other.

Rigid-bodied robot modeling frequently uses long trajectories lasting dozens of seconds, needing control in position mode. These trajectories typically consist of a combination of Fourier series and polynomial functions, helping in parameter identification for the robot dynamic model equations [46]. However, due to their length, torque control is not commonly applied, resulting in suboptimal torque commands for desired movements. To the best of our knowledge, no study has yet investigated the optimal trajectories for data-driven dynamic modeling. Therefore, it is essential to define our set of trajectories to cover a broad range of possible movements.

To facilitate the design of trajectories for the KUKA LBR iiwa robot arm, we establish a working space that defines the area in which our training is focused. This space defines the range of motion for the end-effector and can be re-defined according to the specific tasks for the cobot. In our scenario, the working space boundaries are X = [−0.1,0.1], Y = [0.45,0.65] and Z = [0.3,0.5], with the origin point (0,0,0) defined as the center of robot base. These boundaries position the working space in front of the robot arm, trying to emulate common usage scenarios for this type of robotic arm.

The spiral trajectories (see Fig. 3 Spiral 0, 1 and 2) in these tests were generated using Eq. (11), where Ax_1, Ax_2 and Ax_1 correspond to the chosen axes $[X, Y, Z]$, $[Y, Z, X]$, or $[Z, X, Y]$, depending on the spiral orientation. All three cases are included in the dataset to facilitate movements along any Cartesian axis effectively. Similarly, c_1, c_2 and c_3 denote the center of the trajectory, which is fixed at Cartesian coordinates $x = 0.0, y = 0.55$ and $z = 0.4$. In this equation, T stands for the total number of steps in the trajectory, with $t \in T$ representing the step at time t. The parameter θ determines the number of revolutions the end-effector completes around the center point. More specifically, $\theta = \frac{2\pi}{T/\text{N}^\circ \text{ of revolutions}}$, where "N° of revolutions" denotes the number of revolutions around the center-point.

$$\begin{aligned} Ax_1 &= c_1 + \cos(\theta t) r, \\ Ax_2 &= c_2 + \sin(\theta t) r, \\ Ax_3 &= c_3 + 0.1 \frac{t}{T}, \\ r &= 0.1 \left| t - \frac{T}{2} \right|. \end{aligned} \tag{11}$$

For the rectilinear trajectories (see Fig. 3 Rectilinear 0, 1 and 2), we define a series of key points using Eq. (12), where N represents the

total number of key points and $i \in N$. After determining all the key points, we implement a quintic polynomial interpolation in Cartesian space using the Robotics Toolbox library [47].

Both spiral and rectilinear trajectories undergo changes in size to capture different speeds and variations in speed within the dataset. In the case of spiral trajectories, they initially expand to the maximum radius of the working space before contracting, whereas rectilinear trajectories begin with longer and faster movements to gradually decrease in size towards the midpoint of the trajectory.

$$
\begin{aligned}
Ax_1 &= \begin{cases} c_1 + r & \text{if } i\%4 == 1 \text{ or } i\%4 == 2, \\ c_1 - r & \text{otherwise,} \end{cases} \\
Ax_2 &= \begin{cases} c_2 + r & \text{if } i\%4 > 1, \\ c_2 - r & \text{otherwise,} \end{cases} \\
Ax_3 &= c_3 + 0.1\frac{i}{N}, \\
r &= 0.1 - 0.05\left|i - \frac{N}{2}\right|.
\end{aligned} \tag{12}
$$

In the case of the random trajectories (see Fig. 3 Random 0, 1 and 2), random points are selected within the working space, and then interpolation in joint space is performed using quintic polynomial interpolation.

The number of steps in each trajectory determines the motion speed, significantly affecting the resulting dynamic model. For this reason, the design choice of this parameter will be addressed in Section 3.3.

2.9. Dynamic model learning

To train a ML dynamic model to validate our methodology, the collected data will be organized into tuples $(p_i(t), v_i(t), \tau_i(t))$ representing, position, velocity and torque of joint i at time t, respectively. This arrangement allows the ML dynamic model to infer those torque values per link needed to reach a desired $p_i(t+1)$ and $v_i(t+1)$ based on past and current positions and velocities per link. The collected torque data will be the one extracted from the PD, before gravity compensation is added, as this module is implemented inside the robot system after the torque command is sent through FRI.

The ML model to be used is based on a previous work in [15], consisting of a bidirectional recurrent neural network (BRNN) capable of learning the inverse dynamic model of a cobot. Similarly to the approach followed in [15], we will use the BRNN inverse dynamic model of our cobot as a feedforward controller, integrated with a PD watchdog for feedback support (refer to Section 3.6). Our BRNN will be designed to incorporate the overall dynamic information of the robot, abstracting it from the trajectories obtained with each of the previously tuned PD controllers. This enables the BRNN to operate in a broader workspace without compromising performance, unlike a PD controller tuned for a single working point.

The dynamic dataset will be partitioned into three different subsets: training, validation, and testing. The training subset will be used to train the BRNN inverse dynamic model, while the validation subset will be used to calculate the validation loss during the training process, i.e., the validation loss will indicate when training should be stopped. Subsequently, the test subset will provide an evaluation of the BRNN dynamic model performance. Specifically the training subset will comprise all spiral and rectilinear trajectories, along with the initial random trajectory. The second random trajectory (Random 1) will serve as the validation subset, and the last random trajectory (Random 2) will serve for testing the BRNN performance.

3. Results

3.1. Comparison metrics and methodology

We used the HV as the primary metric to compare the solutions generated by the MOEAs, i.e., NSGA II, SPEA2 and HypE based on the Pareto front for each solution obtained during the tracking of benchmark trajectories. As explained in Section 2.2, the HV calculates the area between the Pareto front, i.e., set of non-dominated solutions found by a MOEA, and a reference point (see Fig. 4.a for a visual representation). This reference point was standardized across all Pareto fronts and selected based on the maximum allowed values for both objective functions. It does not represent a real controller, but rather the maximum values allowed for each objective function. More specifically, the maximum value for the accuracy function was set to a mean average error of 0.03 m (3 cm), and the maximum for the torque function was 0.5. Any point exceeding these values was discarded. Note that the selection of these values can be somewhat arbitrary, as long as they are not too extreme, because they only impact the weight of the extremes of the Pareto front. Using the same reference point for all comparisons ensures that any possible bias introduced by this selection will be shared in all cases, leading to a fairer comparison among algorithms. Due to the stochastic nature of MOEAs, all executions were repeated 5 times to ensure the robustness and reliability of the results. For statistical analyses, we used the *repeated measures ANOVA* test to discern if there are significant differences in our comparisons and *paired t-tests* as Post-hoc tests to identify where those differences lie. To do so, the mean values obtained for each trajectory from Fig. 3 were used for analysis. Later, the difference among algorithms was analyzed between trajectories. This approach allows us to identify significant and consistent differences across all trajectories, thus enabling the selection of the most suitable algorithm with the appropriate hyperparameters. All the settings for the algorithm comparison are detailed in Table 1.

Table 1
Experiment settings for algorithm comparison.

Parameter	Value(s)
N° trials	5
N° of variables	14 (2 per joint)
Max. n° evaluations	4000
HV reference point	[0.03 (MAE), 0.5 (torque function)]
Trajectory length	1000 timesteps (4 s)
N° of trajectories	9

3.2. MOEA selection and fine-tuning

To select the MOEA to be used in the dataset creation, we assessed the performance of three different algorithms, namely NSGA-II [11], SPEA2 [12], and HypE [13]. NSGA-II and SPEA2 were chosen due to their widespread use in multi-objective optimization [48,49]. HypE was also selected as a popular indicator-based MOEA focused on improving our main metric: the HV. We also compared the results of these three algorithms with those of a random search consisting of 4000 evaluations, which served as a baseline for comparison.

All the MOEAs shared the same crossover and mutation operators, and also used the same termination criterion. Due to the encoding used in our problem, i.e., real numbers instead of a binary representation, we decided to use Simulated Binary Crossover (SBX crossover) and polynomial mutation. The mutation probability was set to $\frac{1}{N}$, where N represents the number of variables, i.e., 7 joints with 2 variables per joint, for a total of 14 variables. MOEA comparison was initially performed with a distribution index of 20, a common starting point as observed in [50], but was later fine-tuned to find the optimal value. This adjustment aimed to optimize the performance of these MOEAs for the problem at hand.

Tables A.6 and A.7 in Appendix A contain the results obtained in each trajectory. These tables compare four different population sizes (20, 40, 60 and 80 individuals) since this parameter directly influences both the convergence speed, i.e., number of evaluations, and the final quality of the Pareto front. As observed in Tables A.6 and A.7, a population size of 20 individuals leads to premature convergence, typically

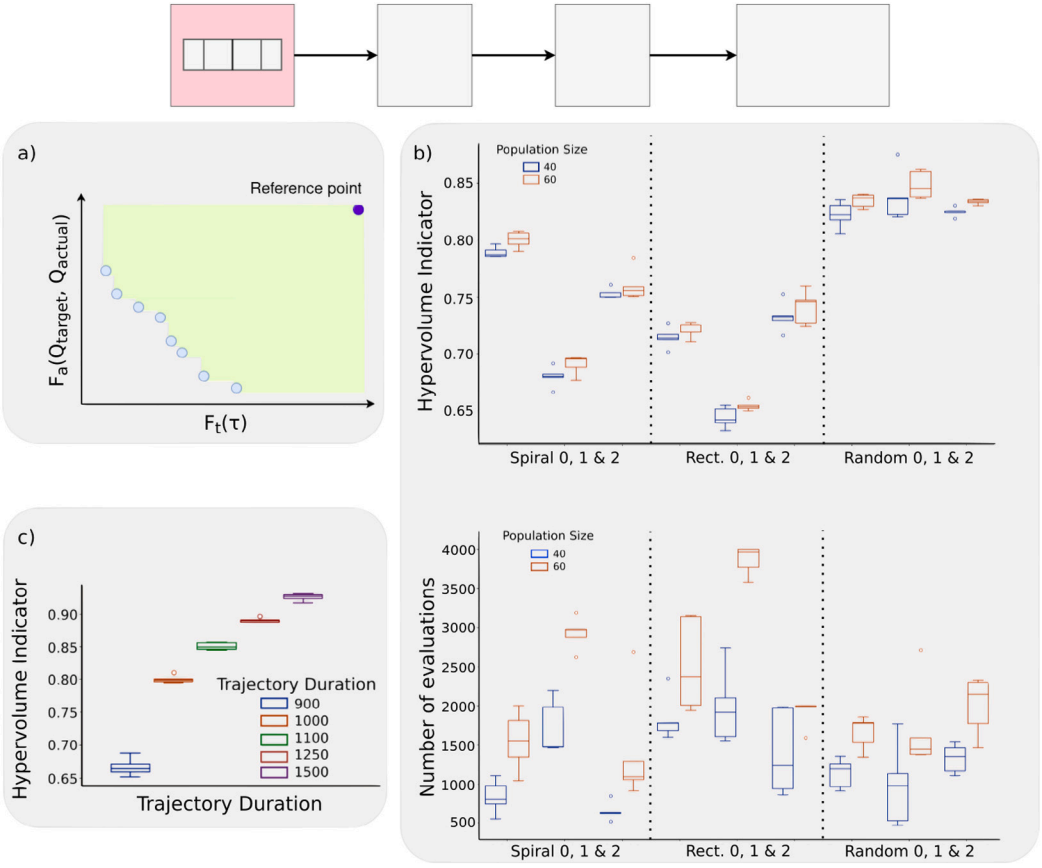


Fig. 4. (a) Diagram depicting the calculation of the hypervolume indicator. This metric is obtained by calculating the area between the Pareto front and a reference point. A higher value indicates a better Pareto front. (b) Differences between the results obtained with a population of size 40 and 60 with the SPEA2 algorithm across all trajectories. We can see that a population of size 60 achieves better Hypervolume values but also takes longer to converge. (c) Comparison when following the Spiral 0 path at different speeds. Trajectory duration is measured in time steps, with each step taking 2 ms. As can be seen, longer durations (slower movements) correspond with higher Hypervolume values, which points to an easier dynamic.

occurring within 500–1000 evaluations. It resulted in similar (or even worse) performance outcomes to the 4000 random evaluations. Hence, this population size was discarded from further consideration. Conversely, when using a population size of 80 individuals, convergence often requires more evaluations, occasionally reaching the maximum allowed limit (4000). Despite this large number of evaluations, the performance outcomes did not exhibit noticeable differences with those generated with a population size of 60 individuals. Consequently, we discard the largest population size as well.

When comparing the three selected MOEAs in Table A.8, we observed no discernible differences among the Pareto fronts, i.e, the HV distribution remains similar across all cases. Nevertheless, when comparing the convergence speed, as indicated in (Tables A.7 and A.9), HypE presents a slower convergence rate compared to NSGA-II and SPEA2, which is even more noticeable with smaller population sizes. Consequently, we discarded HypE for our problem domain. Our analysis did not reveal any significant difference in convergence speed between NSGA-II and SPEA2. The primary difference between these 2 algorithms lies in their chromosome selection, that is, the use of the crowding distance metric in NSGA-II and the neighbor euclidean distance in SPEA2. Nevertheless, due to strict limits on the number of evaluations, these differences do not seem to be reflected in the data. Thus, we used both algorithms in the following sections. Note that despite finding no significant differences in the HV distribution among the 3 MOEAs, we observed a noticeable improvement over the random search whenever the population size was above 20 individuals (with p-values of 0.0006 for NSGA-II, 0.014 for SPEA2 and 0.020 for HypE when the population size is 40, and even smaller values for larger population sizes). This indicates the importance of algorithm selection. However, in the case of our chosen MOEAs, the differences among algorithms, mainly related to the selection of individuals of the same rank, are not significant enough to impact the results of these tests. When comparing population sizes of 40 and 60 individuals, both NSGA-II and SPEA2 show significant differences in both the HV (0.03 and 2.6∗10e−6 respectively) and the number of evaluations (0.001 and 0.0003). This suggests two possible choices: either using a population size of 40 individuals, which will cause less wear and tear on the robot joints at the cost of a slightly worse Pareto front, or using a population size of 60 individuals to achieve optimal results. In our case, we will proceed with a population size of 60 individuals in the following tests.

An example of these differences between population sizes 40 and 60 can be found in Fig. 4.b, where we see the HV and number of evaluations across all trajectories for the SPEA2 algorithm.

As mentioned before, these tests have been performed with a distribution index of 20 for both the mutation and crossover operators, but further tuning of this parameter is advisable [50]. The results of these tests can be seen in Table 2. Only the average of all trajectories is shown in this table for the sake of clarity, but it is important to remember that each trajectory has a different range of values, as seen in Fig. 4.b.

The results of the statistical analysis can be seen in Table 3. When performing a *repeated measures ANOVA*, we find that using the NSGA-II algorithm there are significant differences both in the HV and the number of evaluations (p-values of 0.00 and 0.05, respectively), while SPEA2 only shows a notable difference in HV (p-values of 0.00 and 0.24).

When performing the post-hoc test, we find no difference between values of 1 and 10 in either algorithm. In the same way, a value

Table 2
Algorithm tuning: Distribution index comparison.

Algorithm	Metric	1	10	20	100
NSGA-II	Hypervolume	0.769	0.769	0.765	0.756
	N° evaluations	2185	2177	2231	1897
SPEA2	Hypervolume	0.767	0.767	0.765	0.753
	N° evaluations	2134	2118	2168	1890

Table 3
Algorithm tuning: Statistic analysis of distribution index values.

Algorithm	Metric	ANOVA	1/10 p-val.	10/20 p-val.	20/100 p-val.
NSGA-II	Hypervolume	**0.00**	0.96	0.13	**0.00**
	N° evaluations	**0.05**	0.95	0.69	**0.02**
SPEA2	Hypervolume	**0.00**	0.86	0.23	**0.00**
	N° evaluations	0.24	0.90	0.80	0.04

of 100 seems to underperform in both algorithms but also converge slightly faster. Even though we cannot discard the null-hypothesis when comparing values 10 and 20, a slight trend can be perceived (smaller values seem to work better), especially when looking at Table 2. Thus, it seems advisable to use a distribution index of 10 instead of the default value of 20.

3.3. Analysis of trajectory speed

As discussed in Section 2.8, the length of the benchmark trajectory, that is the number of time steps, significantly influences the robot speed and, consequently, its dynamics and the extracted data. Therefore, we examined the results obtained by applying our optimization algorithms to benchmark trajectories while varying their temporal duration. Fig. 4.c illustrates the results obtained from the spiral path (Spiral 0) depicted in Fig. 3. These trajectories follow identical paths but vary in the number of time steps, ranging from 900 steps (equivalent to 3.6 s, resulting in faster motions) to 1500 steps (equivalent to 6 s, resulting in slower motions). Note that we use always 250 Hz, thus 250 steps per second.

As depicted in Fig. 4.c, slower trajectories are easier to track, leading to an increase in the HV. This reflects the fact that slower trajectories show simpler and more linear dynamics, while faster movements, characterized by more ballistic motion, give rise to more complex dynamics with stronger nonlinear effects. Note that this effect is not linear and, consequently, the difference between a trajectory of 900 time-steps and one of 1000 is much higher than the difference between 1000 and 1100. When attempting to execute trajectories in 800 time-steps, the values of the objective functions typically fall outside of the values chosen as the reference point (MAE of 0.03 and torque function of 0.5), that is, trajectories become too risky to optimize. This means that the data extracted during dataset creation should be well-matched and coordinated to the task to be performed by the final controller. In some cases, it may even be advisable to include different speeds for each path during the data extraction phase.

3.4. Comparison between generic and fine-tuned PD

To illustrate the necessity of fine-tuning PD controllers for each trajectory individually, and to highlight the challenge of achieving a universal control system applicable to various trajectories, a set of "generic controllers" underwent tuning using the NSGA-II algorithm with a population size of 60 and a distribution index of 10. Each controller was evaluated using an average of every trajectory. Hence, if a single PD controller could perform well on every trajectory, or if the designed trajectories were too similar, we would expect similar performance between the generic controllers and those fine-tuned using only one trajectory each. Nevertheless, Table 4 illustrates a significant difference in performance between the general and fine-tuned controllers. This difference is particularly noticeable when attempting to balance the torque value and accuracy, as depicted in Fig. 5.a where the biggest difference is observed in the central region, while the extremes show a closer alignment.

Table 4
Comparison between general and fine-tuned PD. Hypervolume Indicator is used to compare Pareto fronts. A higher value indicates a better Pareto front.

Trajectory	General	Fine-tuned
Spiral 0	0.779 ± 0.008	**0.806 ± 0.015**
Spiral 1	0.652 ± 0.007	**0.693 ± 0.012**
Spiral 2	0.738 ± 0.007	**0.759 ± 0.005**
Rectilinear 0	0.7 ± 0.006	**0.729 ± 0.013**
Rectilinear 1	0.599 ± 0.016	**0.646 ± 0.013**
Rectilinear 2	0.73 ± 0.008	**0.746 ± 0.016**
Random 0	0.792 ± 0.012	**0.835 ± 0.005**
Random 1	0.835 ± 0.009	**0.847 ± 0.011**
Random 2	0.823 ± 0.003	**0.839 ± 0.003**

3.5. Decision making: Controller selection

Once the final Pareto front is obtained, selecting one controller for each trajectory becomes crucial to avoid conflicting data. To that aim, different approaches to controller selection exist, as documented in [48]. In our case, an *a posteriori* preference method was used, that is, the selection process occurred after the search concluded.

Many different approaches to controller selection are plausible, such as measuring the distance of each point to a desired optimum [51]. However, determining this optimum value beforehand is typically not feasible in a real-world setting such as the one being emulated here. Alternatively, another possible approach is to use a utility function to weight each objective [52]. However, note that accurately weighting each objective can be challenging, which is one of the primary reasons for using a multi-objective algorithm.

In this study, we want to emphasize the knee-based approach, which prioritizes finding the optimal trade-off between objectives [53–55]. There are different methods for defining a knee in a Pareto front, such as measuring the angle between nearby points [53] or specifying a desired trade-off between objectives [54]. While these methods often identify multiple points of interest requiring expert evaluation, this study favors those approaches that yield a single optimal value, such as the normal boundary intersection or line-distance-based method [56]. This method involves drawing a line between the two extremes of the Pareto front and calculating the perpendicular distance towards the origin of each non-dominated solution to this line (see Fig. 5.b). This method shares similarities with the Hypervolume-based approach, particularly when the HV reference point for calculating the dominated area is based on the maximum values of the Pareto front in each objective.

3.6. Use case example; the BRNN controller within a feedforward control loop

To demonstrate the usefulness of storing dynamic robot information extracted from MOEA optimal controllers into a dataset for training any ML controller, we trained different BRNN-based dynamic models of the robot using trajectories obtained from a tuned PD controller as in [15]. Moreover, to evaluate the influence of dataset quality on the BRNN-based dynamic model, we trained two models: (a) using data extracted with generic PD controllers, and (b) using data extracted with fine-tuned, trajectory-specific PD controllers (see Section 3.4). Both models were trained with data divided identically and the same configuration, including hyperparameters.

The two BRNN-based dynamic models consisted of 64 GRU units and used a temporal window of 97 time steps, accounting for time

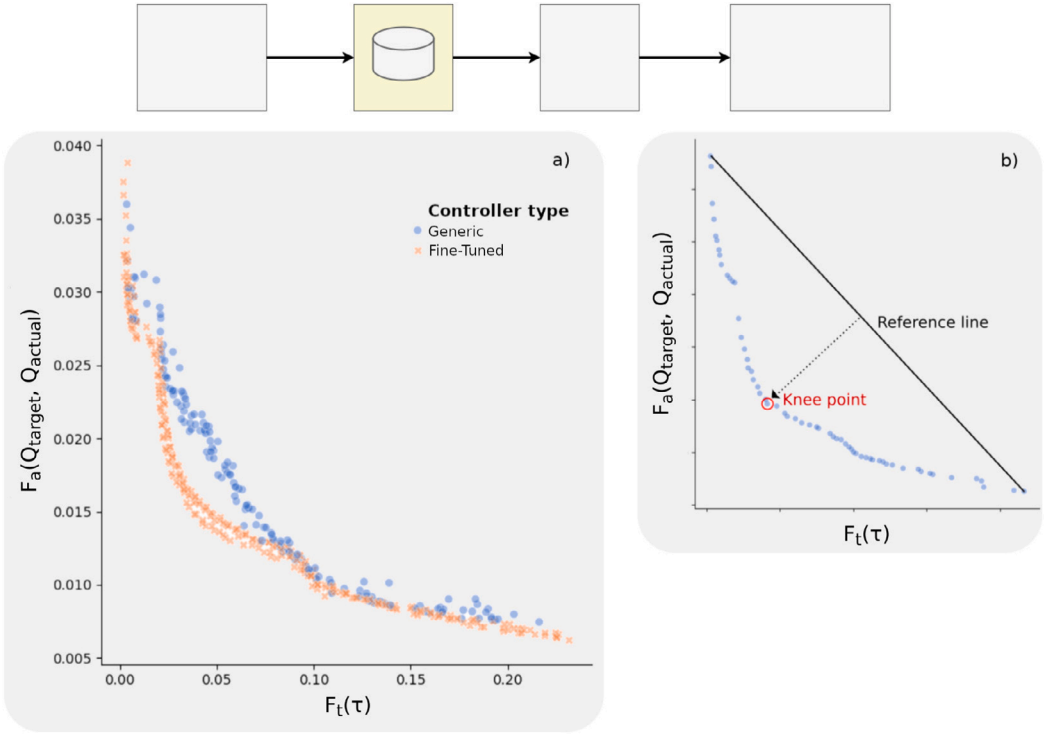


Fig. 5. (a) Comparison between generic and fine-tuned controllers. The generic controllers have been tuned using all the trajectories in the dataset, while the fine-tuned ones only use the specific trajectory in which they are being tested. (b) Diagram depicting the calculation of the line-based knee. A line is drawn between the extremes of the Pareto front and the perpendicular distance between said line and each non-dominated solution is used to find the knee-point. The trajectory shown in both cases is Spiral 1.

steps [t−48, t+48]. During the training stage, values corresponding to future time steps were known in advance; however, during the inference stage, they were substituted with the desired trajectory values. The BRNN-based model trained with data extracted from fine-tuned, trajectory-specific PD controllers achieved an R^2 value of 0.94 on the test set, indicating high prediction accuracy, whereas the model trained with data from generic PD controllers achieved a lower prediction accuracy, with an R^2 value of 0.87. Both BRRN-based models were used as feedforward controllers, both following the non-parametric inverse dynamic configuration (NID) architecture described in [15]. The feedforward controller was used conjointly with a feedback PD controller to prevent drift from the actual trajectory, i.e, as a watch-dog or complementary safety controller only becoming significantly active in case of major deviations. The resulting control loop is depicted in Fig. 7.a.

For tuning the feedback PD watch-dog controller, we followed the same steps as for the generic PD controllers described in Section 3.4. We used lower upper limits for each variable to be tuned, thereby limiting their overall control actions, as the main contributor in the control loop should be the feedforward controller. The upper limits used were KP = {320, 320, 176, 176, 110, 40, 40}, KD = {1.28, 1.28, 0.7, 0.7, 0.44, 0.16, 0.16} (five times lower than in Section 2.2).

To demonstrate the dominance of feedforward control action, we compared the Torque Time Integral (TTI) provided by both the feedback (PD) and feedforward (BRNN model) controllers for each trajectory (see Table 5). The TTI offers a clear measure of the energy output by each controller, revealing that the feedforward controller had a significantly greater impact on the overall control system output. In contrast, the PD watchdog controller primarily corrected minor deviations that occurred during the trajectory execution.

The influence of data quality within the dynamic dataset on the subsequent performance of the BRNN-based controller was clearly illustrated when comparing the MOEA Pareto front obtained in four different scenarios in torque-control trajectory-tracking: (1) feedback control loop using generic PD controllers, (2) feedback control loop using fine-tuned, trajectory-specific PD controllers, (3) feedforward + watch-dog PD control loop using a BRNN-based inverse dynamic model trained with data extracted from generic PD controllers, and (4) feedforward + watch-dog PD control loop using a BRNN-based inverse dynamic model trained with data extracted with fine-tuned, trajectory-specific PD controllers. Cases 1 and 2 are extracted from the Pareto fronts shown in Section 3.4. The Pareto front results (see Fig. 7.b) revealed the advantage of using a dataset from which an inverse data-based dynamic model could be extracted, as in cases 3 and 4, where the Pareto front consistently outperformed the best results obtained for cases 1 and 2. When comparing cases 3 and 4, the results confirmed the advantage of having quality data for extracting the data-based inverse dynamic model for the robot. There was a clear improvement achieved when using fine-tuned data, particularly noticeable in the accuracy function ($F_a(Q_{target}, Q_{actual})$), resulting in a lower mean average error for the same torque function (See Fig. 6).

Table 5
Torque Time Integral (TTI) in N m*s provided by both the feedback (PD) and feedforward controllers (BRNN-based model trained using data extracted from fine-tuned, trajectory-specific PD controllers) for each trajectory. The feedforward component torque contributes significantly more to the overall torque control action within the control loop.

Trajectory	PD watch-dog	Feedforward model
Spiral 0	2145 ± 185	**13201 ± 103**
Spiral 1	2920 ± 236	**10026 ± 47**
Spiral 2	2394 ± 97	**15022 ± 89**
Rectilinear 0	4194 ± 330	**9849 ± 41**
Rectilinear 1	4569 ± 166	**9443 ± 114**
Rectilinear 2	4426 ± 153	**10353 ± 109**
Random 0	2711 ± 116	**7402 ± 60**
Random 1	3484 ± 230	**7587 ± 29**
Random 2	3128 ± 82	**6458 ± 110**

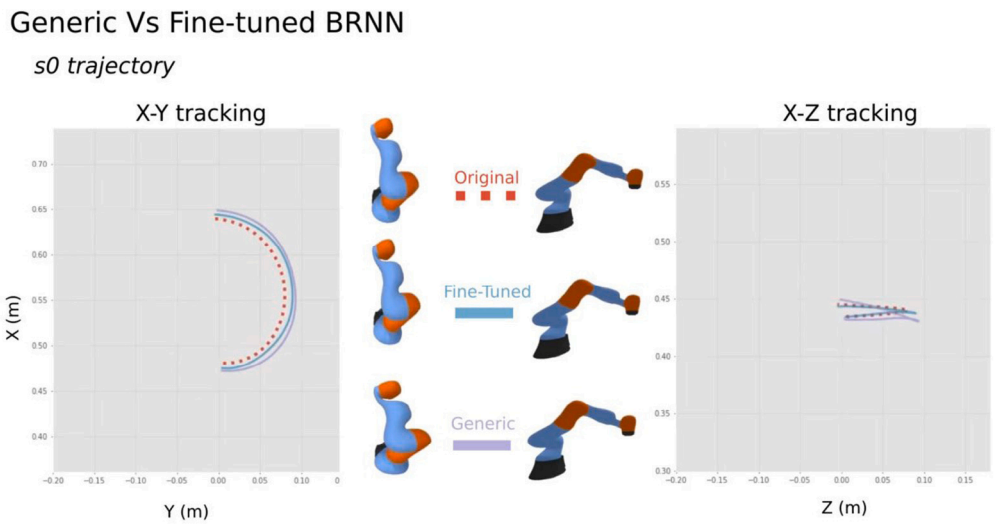


Fig. 6. Snapshot from the video comparing the accuracy of a BRNN model trained with fine-tuned data and one trained using generic PD data. The video can be found at https://youtu.be/JKbuu6ykNJs.

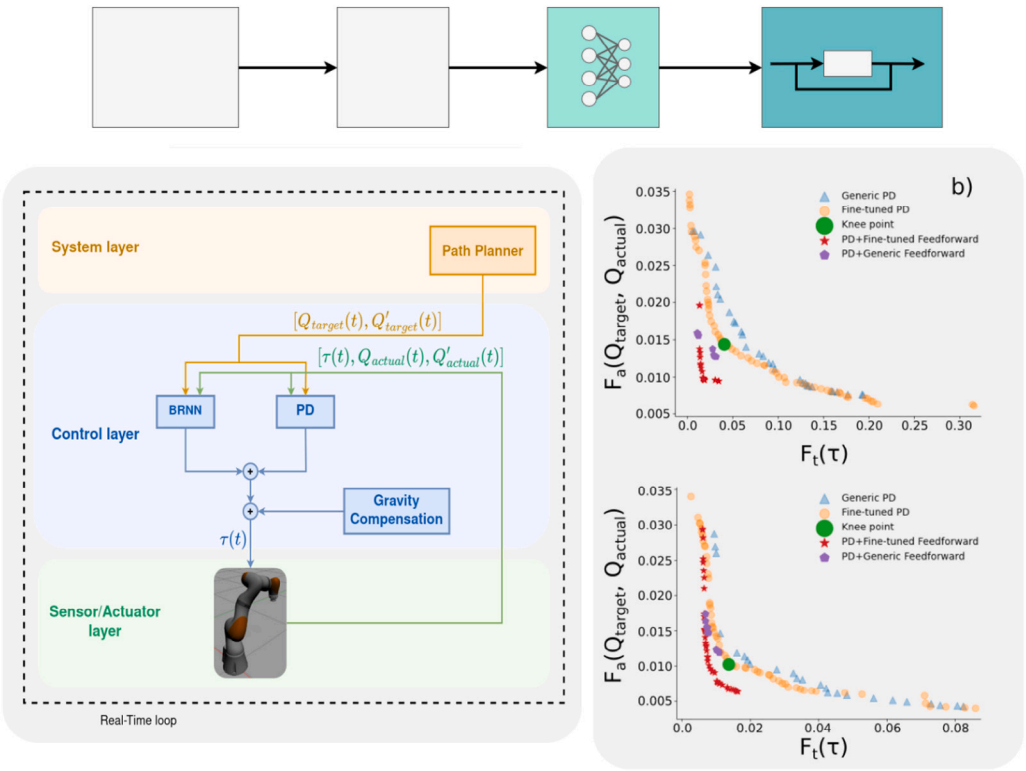


Fig. 7. (a) Flow diagram of the Feedforward + watch-dog PD torque control loop. (b) Comparison of the MOEA Pareto front obtained in four different scenarios in torque-control trajectory-tracking: (blue triangle) feedback control loop using generic PD controllers, (yellow circle) feedback control loop using fine-tuned, trajectory-specific PD controllers, (purple pentagon) feedforward + watch-dog PD control loop using a BRNN-based inverse dynamic model trained with data extracted from generic PD controllers, and (red star) feedforward + watch-dog PD control loop using a BRNN-based inverse dynamic model trained with data extracted with fine-tuned, trajectory-specific PD controllers. Tracked trajectories depicted are: Spiral 1 (up), which was part of the training set, and Random 2 (down), which was only used in the test set. The knee point used for the fine-tuned dataset creation is marked for an easier comparison of the improvements over the original PD controller. The set of controllers obtained using feedforward BRNN inverse dynamic models improves upon the results obtained by the fine-tuned PDs, and a clear enhancement is observed when training the BRNN-based inverse dynamic model with data extracted from the fine-tuned PD controllers instead of generic PD controllers.

4. Conclusions

The presented work addresses challenges in creating a dataset for data-driven dynamic model identification of a collaborative robot. This requires the use of pre-existing controllers that can be fine-tuned to different trajectories, facilitating the gathering of richer samples for the dataset. As demonstrated, this fine-tuning can be achieved by combining PD control and MOEAs to ensure both accuracy and safety in generated motions. To evaluate this PD-MOEAs combination, we conducted tests comparing NSGA-II, the state of the art MOEA used in most robotic PID applications [48] with two similar MOEAs, SPEA2 and HypE.

We have also demonstrated the potential of constructing a dynamic dataset designed for highly nonlinear dynamic systems, such as cobots.

Table A.6
Algorithm comparison: Hypervolume indicator.

Algorithm	Pop. size	Spiral 0	Spiral 1	Spiral 2	Rect. 0	Rect. 1
Random	4000	0.78 ± 0.004	0.66 ± 0.003	0.74 ± 0.006	0.68 ± 0.005	0.59 ± 0.012
NSGA-II	20	0.77 ± 0.011	0.65 ± 0.029	0.73 ± 0.015	0.68 ± 0.027	0.61 ± 0.028
	40	0.78 ± 0.019	0.68 ± 0.004	0.75 ± 0.015	0.71 ± 0.012	0.63 ± 0.005
	60	0.80 ± 0.020	0.68 ± 0.018	0.75 ± 0.006	0.72 ± 0.007	0.64 ± 0.012
	80	0.79 ± 0.017	0.69 ± 0.008	0.76 ± 0.009	0.72 ± 0.005	0.64 ± 0.010
SPEA2	20	0.77 ± 0.023	0.66 ± 0.026	0.73 ± 0.015	0.70 ± 0.019	0.61 ± 0.014
	40	0.78 ± 0.006	0.68 ± 0.011	0.75 ± 0.006	0.71 ± 0.011	0.64 ± 0.011
	60	0.80 ± 0.009	0.69 ± 0.010	0.76 ± 0.018	0.72 ± 0.008	0.65 ± 0.005
	80	0.81 ± 0.023	0.69 ± 0.006	0.75 ± 0.009	0.71 ± 0.009	0.64 ± 0.018
HypE	20	0.77 ± 0.012	0.66 ± 0.010	0.74 ± 0.007	0.67 ± 0.047	0.63 ± 0.018
	40	0.78 ± 0.009	0.68 ± 0.011	0.75 ± 0.011	0.71 ± 0.007	0.63 ± 0.016
	60	0.79 ± 0.020	0.69 ± 0.013	0.76 ± 0.017	0.71 ± 0.016	0.64 ± 0.010
	80	0.79 ± 0.013	0.69 ± 0.008	0.76 ± 0.022	0.72 ± 0.011	0.65 ± 0.020

Algorithm	Pop. size	Rect. 2	Random 0	Random 1	Random 2
Random	4000	0.72 ± 0.003	0.82 ± 0.005	0.83 ± 0.004	0.82 ± 0.001
NSGA-II	20	0.73 ± 0.013	0.81 ± 0.016	0.83 ± 0.023	0.82 ± 0.015
	40	0.75 ± 0.013	0.83 ± 0.008	0.85 ± 0.011	0.83 ± 0.005
	60	0.75 ± 0.016	0.83 ± 0.009	0.86 ± 0.013	0.84 ± 0.004
	80	0.75 ± 0.017	0.83 ± 0.010	0.85 ± 0.010	0.84 ± 0.004
SPEA2	20	0.72 ± 0.012	0.76 ± 0.077	0.82 ± 0.013	0.82 ± 0.003
	40	0.73 ± 0.016	0.82 ± 0.014	0.84 ± 0.037	0.83 ± 0.005
	60	0.74 ± 0.018	0.83 ± 0.008	0.85 ± 0.016	0.83 ± 0.003
	80	0.76 ± 0.018	0.84 ± 0.006	0.85 ± 0.016	0.84 ± 0.006
HypE	20	0.73 ± 0.037	0.82 ± 0.009	0.84 ± 0.017	0.82 ± 0.011
	40	0.74 ± 0.014	0.82 ± 0.019	0.85 ± 0.023	0.82 ± 0.017
	60	0.76 ± 0.026	0.83 ± 0.004	0.85 ± 0.021	0.83 ± 0.006
	80	0.76 ± 0.011	0.83 ± 0.012	0.85 ± 0.010	0.83 ± 0.008

We include the obtained results of the different MOEAs for 9 different trajectories.
* A higher value represents a better set of solutions.

This dataset enables a BRNN operating as a controller to capture nonlinear behaviors comprehensively within a data-driven inverse dynamic model. This capability increases the overall performance of the BRNN controller compared to generic PD solutions in trajectory tracking at low torque conditions. The combination of BRNN feedforward control and PD feedback acting as a watch-dog is shown to outperform even the fine-tuned PDs from which the original data was extracted.

Moreover, implementing this PD-MOEAs combination methodology into a real system presents new challenges. Simulation dynamics are often too different from the real system to allow for direct sim-to-real tuning, meaning that the PD parameters obtained in a simulated environment may not be directly applicable to a real robot. For this reason, we recommend tuning the controllers directly using real hardware, with a focus on convergence speed and the number of evaluations to minimize wear and tear of the cobot joints. Additional challenges may arise from controlling MOEA randomization to ensure robot and human safety during the fine-tuning process, as some intermediate PD combinations could result in unsafe motions (with risks of damage to the different robot components).

CRediT authorship contribution statement

Diego Navarro-Cabrera: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Juan H. García-Guzmán:** Software, Investigation. **Nicolás C. Cruz:** Writing – review & editing, Validation, Supervision. **Brayan Valencia-Vidal:** Writing – review & editing, Software, Data curation. **Niceto R. Luque:** Writing – review & editing, Supervision, Project administration, Funding acquisition. **Eduardo Ros:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was supported by the EU with the IMOCOe4.0 [EU H2020RIA-101007311] project and by Spanish national funding [PCI2021-121925]. This project has received funding from the ECSEL Joint Undertaking (JU) under Grant Agreement No 101007311. This study was also supported by SPIKEAGE [PID2020-113422GA-I00] by the Spanish Ministry of Science and Innovation MCIN/AEI/ 10.13039/501100011033, awarded to NRL; DLROB [TED2021-131294B-I00] funded by MCIN/AEI/ 10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, awarded to NRL; MUSCLEBOT [CNS2022-135243] funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, awarded to NRL; Grant PID2022-140095NB-I00 funded by MICIU/AEI/ 10.13039/501100011033/ and FEDER, UE. Grant PID2022-140095NB-I00 funded by MICIU/AEI/ 10.13039/501100011033/ and FEDER, UE awarded to ER. N.C. Cruz is supported by the Ministry of Economic Transformation, Industry, Knowledge and Universities from the Andalusian government (PAIDI 2021: POSTDOC_21_00124).

Appendix A. Algorithm comparison tables

See Tables A.6–A.9.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2025.105006.

Table A.7
Algorithm comparison: Number of evaluations..

Algorithm	Pop. size	Spiral 0	Spiral 1	Spiral 2	Rect. 0	Rect. 1
NSGA-II	20	654 ± 414	691 ± 146	336 ± 113	816 ± 493	785 ± 266
	40	836 ± 302	1726 ± 698	1145 ± 377	1529 ± 557	1571 ± 389
	60	1571 ± 560	3397 ± 937	1272 ± 383	2729 ± 493	3270 ± 1000
	80	2072 ± 1091	3680 ± 405	1645 ± 639	3430 ± 664	3099 ± 808
SPEA2	20	568 ± 228	826 ± 296	357 ± 126	857 ± 323	946 ± 197
	40	837 ± 265	1714 ± 429	650 ± 146	1834 ± 365	1981 ± 594
	60	1547 ± 469	2923 ± 254	1406 ± 905	2518 ± 737	3859 ± 229
	80	2257 ± 721	4000 ± 0	1539 ± 374	2911 ± 1066	3457 ± 936
HypE	20	598 ± 216	1045 ± 335	337 ± 41	778 ± 682	1367 ± 344
	40	1025 ± 578	2374 ± 977	626 ± 167	2151 ± 605	2334 ± 1379
	60	1176 ± 911	3560 ± 855	1254 ± 403	3092 ± 887	3943 ± 157
	80	1720 ± 769	4000 ± 0	1439 ± 925	3865 ± 372	3600 ± 1110

Algorithm	Pop. size	Rect. 2	Random 0	Random 1	Random 2
NSGA-II	20	863 ± 326	572 ± 109	374 ± 260	539 ± 194
	40	1619 ± 385	1129 ± 103	1260 ± 465	1329 ± 369
	60	1923 ± 138	1964 ± 307	1829 ± 390	2191 ± 587
	80	3217 ± 777	2093 ± 942	1904 ± 478	2674 ± 685
SPEA2	20	627 ± 397	644 ± 165	378 ± 148	568 ± 172
	40	1394 ± 678	1132 ± 236	1101 ± 814	1322 ± 229
	60	1914 ± 226	1747 ± 308	1694 ± 711	1999 ± 463
	80	3313 ± 859	2619 ± 502	2037 ± 728	2466 ± 729
HypE	20	855 ± 594	711 ± 280	516 ± 506	741 ± 319
	40	1709 ± 486	1416 ± 196	1280 ± 565	1322 ± 506
	60	3192 ± 690	2055 ± 460	1581 ± 693	2149 ± 1032
	80	4000 ± 0	2057 ± 578	2351 ± 294	1989 ± 275

We include the obtained results of the different MOEAs for 9 different trajectories.
* Maximum number of evaluations allowed is 4000.

Table A.8
Algorithm comparison: Statistic analysis of the Hypervolume Indicator.

Pop. size	ANOVA	NSGA-II/SPEA2		NSGA-II/HypE		SPEA2/HypE	
	<0.05	*p*-value	Statistics	*p*-value	Statistics	*p*-value	Statistics
20	0.32	0.45	0.77	0.16	−1.54	0.26	−1.19
40	0.61	0.48	0.73	0.40	0.87	0.92	0.09
60	0.90	0.77	0.29	0.84	−0.20	0.68	−0.42
80	0.84	0.64	−0.47	0.28	−1.15	0.94	0.06

Table A.9
Algorithm comparison: Statistic analysis of the number of evaluations.

Pop. size	ANOVA	NSGA-II/SPEA2		NSGA-II/HypE		HypE/SPEA2	
	<0.05	*p*-value	Statistics	*p*-value	Statistics	*p*-value	Statistics
20	**0.03**	0.70	−0.39	0.07	−2.08	**0.03**	**−2.58**
40	**0.04**	0.83	0.12	0.11	−1.73	**0.00**	**−3.70**
60	0.20	0.56	0.60	0.25	−1.21	0.15	−1.58
80	0.67	0.43	−0.81	0.42	−0.84	0.80	−0.26

Data availability

Data will be made available on request.

References

[1] Shirine El Zaatari, Mohamed Marei, Weidong Li, Zahid Usman, Cobot programming for collaborative industrial tasks: An overview, Robot. Auton. Syst. 116 (2019) 162–180.

[2] Hicham Chaoui, Pierre Sicard, Wail Gueaieb, ANN-based adaptive control of robotic manipulators with friction and joint elasticity, IEEE Trans. Ind. Electron. 56 (2009) 3174–3187.

[3] Sami Haddadin, Physical safety in robotics, in: SyDe Summer School, 2015.

[4] Rongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin, Birgitta Dresp-Langley, Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review, Robot. 10 (1) (2021).

[5] I. Elguea-Aguinaco, Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Ibai Inziarte-Hidalgo, Simon Bøgh, Nestor Arana-Arexolaleiba, A review on reinforcement learning for contact-rich robotic manipulation tasks, Robot. Comput.-Integr. Manuf. 81 (2023) 102517.

[6] Yingbai Hu, Mingyang Cui, Jianghua Duan, Wenjun Liu, Dianye Huang, Alois Knoll, Guang Chen, Model predictive optimization for imitation learning from demonstrations, Robot. Auton. Syst. 163 (2023) 104381.

[7] Andrew S. Robbins, Miao Ling Ho, M. Teodorescu, Model-free dynamic control of robotic joints with integrated elastic ligaments, Robot. Auton. Syst. 155 (2022) 104150.

[8] Mei Liu, Yutong Li, Yingqi Chen, Yimeng Qi, Long Jin, A distributed competitive and collaborative coordination for multirobot systems, IEEE Trans. Mob. Comput. 23 (2024) 11436–11448.

[9] N.D. Vuong, M.H. Ang, Dynamic model identification for industrial robots, Acta Polytech. Hung. 6 (5) (2009) 51–68.

[10] Rafael Kelly, PD control with desired gravity compensation of robotic manipulators, Int. J. Robot. Res. 16 (1997) 660–672.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.

[12] Eckart Zitzler, Marco Laumanns, Lothar Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization, 3242, 2001,

[13] Johannes Bader, Eckart Zitzler, Hype: An algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (2011) 45–76.

[14] Sajid Mohamed, Gijs van der Veen, Hans Kuppens, Matias Vierimaa, Tassos Kanellos, Henry Stoutjesdijk, Riccardo Masiero, Kalle Määttä, Jan Wytze van der Weit, Gabriel Ribeiro, Ansgar Bergmann, Davide Colombo, Javier Arenas, Alfie Keary, Martin Goubej, Benjamin Rouxel, Pekka Kilpeläinen, Roberts Kadiķis, Mikel Armendia, Petr Blaha, Joep Stokkermans, Martin Cech, Arend-Jan Beltman, The IMOCO4.e reference framework for intelligent motion control systems, 2023 IEEE 28th Int. Conf. Emerg. Technol. Fact. Autom. (ETFA) (2023) 1–8.

[15] Brayan Valencia-Vidal, Eduardo Ros, Ignacio Abadía, Niceto R. Luque, Bidirectional recurrent learning of inverse dynamic models for robots with elastic joints: a real-time real-world implementation, Front. Neurorobotics 17 (2023).

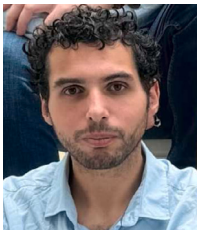
[16] Helon Vicente Hultmann Ayala, Leandro dos Santos Coelho, Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator, Expert Syst. Appl. 39 (10) (2012) 8968–8974.

[17] Liu Qiang, Shi Xuhua, Lan Ting, Chen Xiaoxia, Zhuang Jianpei, Multi-objective optimization based self tuning robot manipulator controller, 2019 Chin. Control. Decis. Conf. (CCDC) (2019) 2593–2598.

[18] Vijay Mohan, Himanshu Chhabra, Asha Rani, Vijander Singh, Robust self-tuning fractional order PID controller dedicated to non-linear dynamic system, J. Intell. Fuzzy Syst. 34 (2018) 1467–1478.
[19] Vijay Mohan, Himanshu Chhabra, Asha Rani, Vijander Singh, An expert 2DOF fractional order fuzzy PID controller for nonlinear systems, Neural Comput. Appl. 31 (2019) 4253–4270.
[20] Vijay Mohan, Bharti Panjwani, Himanshu Chhabra, Asha Rani, Vijander Singh, Self-regulatory fractional fuzzy control for dynamic systems: An analytical approach, Int. J. Fuzzy Syst. 25 (2022) 794–815.
[21] Himanshu Chhabra, Vijay Mohan, Asha Rani, Vijander Singh, Robust nonlinear fractional order fuzzy PD plus fuzzy I controller applied to robotic manipulator, Neural Comput. Appl. 32 (2019) 2055–2079.
[22] Eloise Matheson, Riccardo Minto, Emanuele G.G. Zampieri, Maurizio Faccio, Giulio Rosati, Human-robot collaboration in manufacturing applications: A review, Robotics 8 (2019) 100.
[23] KU.K.A. Robotics Corporation, LBR iiwa, 2024, URL https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa.
[24] Nicolás C. Cruz, Juana López Redondo, Eva M. Ortigosa, Pilar Martínez Ortigosa, On the design of a new stochastic meta-heuristic for derivative-free optimization, in: Communication Systems and Applications, 2022.
[25] G. Lindfield, John Penny, Introduction to nature-inspired optimization, 2017, pp. 1–238,
[26] Alberto Costa, Giacomo Nannicini, Rbfopt: an open-source library for black-box optimization with costly function evaluations, Math. Program. Comput. 10 (2018).
[27] Sad Salhi, Heuristic search: The emerging science of problem solving, 2017, URL https://doi.org/10.1007/978-3-319-49355-8.
[28] Milagros Marín, María José Sáez-Lara, Eduardo Ros, Jesús Alberto Garrido, Optimization of efficient neuron models with realistic firing dynamics. the case of the cerebellar granule cell, Front. Cell. Neurosci. 14 (2020).
[29] Milagros Marín, Nicolás C. Cruz, Eva M. Ortigosa, María José Sáez-Lara, Jesús Alberto Garrido, Richard R. Carrillo, On the use of a multimodal optimizer for fitting neuron models. application to the cerebellar granule cell, Front. Neuroinformatics 15 (2021).
[30] Algirdas Lancinskas, Pilar Ortigosa, Julius Zilinskas, Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm, Nonlinear Anal. Model. Control 18 (2013) 293–313.
[31] Ernestas Filatovas, Algirdas Lancinskas, Olga Kurasova, Julius Zilinskas, A preference-based multi-objective evolutionary algorithm R-NSGA-ii with stochastic local search, Central Eur. J. Oper. Res. 25 (2016) 859–878.
[32] Ignacio Abadía, Francisco Naveros, Jesús Alberto Garrido, Eduardo Ros, Niceto R. Luque, On robot compliance: A cerebellar control approach, IEEE Trans. Cybern. 51 (2020) 2476–2489.
[33] Saeed Tavakoli, Ian Griffin, Peter John Fleming, Multi-objective optimization approach to the PI tuning problem, 2007 IEEE Congr. Evol. Comput. (2007) 3165–3171.
[34] Andreia P. Guerreiro, Carlos M. Fonseca, Luís Paquete, The hypervolume indicator: Computational problems and algorithms, ACM Comput. Surv. 54 (6) (2021).
[35] Katsutoshi Tamura, Shinji Miura, Necessary and sufficient conditions for local and global nondominated solutions in decision problems with multi-objectives, J. Optim. Theory Appl. 28 (1979) 501–523.
[36] Lei Troy Hao, Roberto Pagani, Manuel Beschi, Giovanni Legnani, Dynamic and friction parameters of an industrial robot: Identification, comparison and repetitiveness analysis, Robot. 10 (2021) 49.
[37] Jianwei Dong, Jianming Xu, Qiaoqian Zhou, Junwei Zhu, Li Yu, Dynamic identification of industrial robot based on nonlinear friction model and LS-SOS algorithm, IEEE Trans. Instrum. Meas. 70 (2021) 1–12.
[38] Chan Lee, Suhui Kwak, Jihoo Kwak, Sehoon Oh, Generalization of series elastic actuator configurations and dynamic behavior comparison, Actuators 6 (2017) 26.
[39] Ignacio Abadía, Francisco Naveros, Eduardo Ros, Richard R. Carrillo, Niceto R. Luque, A cerebellar-based solution to the nondeterministic time delay problem in robotic control, Sci. Robot. 6 (2021).
[40] Yazhou Hu, Wenxue Wang, Hao Liu, Lianqing Liu, Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model, IEEE Trans. Neural Networks Learn. Syst. 31 (2020) 3570–3578.
[41] Maciej Bednarczyk, ICube-robotics/iiwaros2: ros-galactic, 2022, http://dx.doi.org/10.5281/zenodo.6420784, URL https://doi.org/10.5281/zenodo.6420784.
[42] Nathan P. Koenig, Andrew Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, 2004 IEEE/ RSJ Int. Conf. Intell. Robot. Syst. (IROS) (IEEE Cat. No. 04CH37566) 3 (2004) 2149–2154 vol.3.
[43] Martin Cech, Arend-Jan Beltman, Kaspars Ozols, Digital twins and AI in smart motion control applications, 2022 IEEE 27th Int. Conf. Emerg. Technol. Fact. Autom. (ETFA) (2022) 1–7.
[44] Steve Macenski, Tully Foote, Brian P. Gerkey, Chris Lalancette, William Woodall, Robot operating system 2: Design, architecture, and uses in the wild, Sci. Robot. 7 (2022).
[45] Giuseppe Carlo Calafiore, Marina Indri, Basilio Bona, Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation, J. Field Robot. 18 (2001) 55–68.
[46] Tolga-Can Callar, Sven Böttger, Hybrid learning of time-series inverse dynamics models for locally isotropic robot motion, IEEE Robot. Autom. Lett. 8 (2022) 1061–1068.
[47] Peter Corke, Jesse Haviland, Not your grandmother's toolbox–the robotics toolbox reinvented for python, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 11357–11363.
[48] Alejandro Rodríguez-Molina, Efrén Mezura-Montes, Miguel Gabriel Villarreal-Cervantes, Mario Aldape-Pérez, Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem, Appl. Soft Comput. 93 (2020) 106342.
[49] Yunfei Cui, Zhiqiang Geng, Qunxiong Zhu, Yongming Han, Review: Multi-objective optimization methods and application in energy saving, Energy 125 (2017) 681–704.
[50] Mohammad Hamdan, Revisiting the distribution index in simulated binary crossover operator for evolutionary multiobjective optimisation algorithms, 2014 Fourth Int. Conf. Digit. Inf. Commun. Technol. Appl. (DICTAP) (2014) 37–41.
[51] Eberhard E. Bischoff, A posteriori trade-off analysis in reference point approaches, 1984, URL https://doi.org/10.1007/978-3-662-00184-4_16.
[52] Jianbo Yang, Pratyush Sen, Preference modelling by estimating local utility functions for multiobjective optimization, European J. Oper. Res. 95 (1996) 115–138.
[53] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, Matthias Osswald, Finding knees in multi-objective optimization, in: Parallel Problem Solving from Nature, 2004.
[54] Kalyanmoy Deb, Shivam Gupta, Understanding knee points in bicriteria problems and their implications as preferred solution principles, Eng. Optim. 43 (2011) 1175–1204.
[55] Arash Heidari, Jixiang Qing, Sebastian Rojas-Gonzalez, Jürgen Branke, Tom Dhaene, Ivo Couckuyt, Finding knees in Bayesian multi-objective optimization, in: Parallel Problem Solving from Nature, 2022.
[56] Indraneel Das, On characterizing the "knee" of the Pareto curve based on normal-boundary intersection, Struct. Optim. 18 (1999) 107–115.

Diego Navarro-Cabrera Diego Navarro Cabrera is a PhD student who graduated from Computer Science in 2021 and got a Masters Degree in Data Science in 2022, both at University of Granada. During his Masters Degree he joined the Applied Computational Neuroscience (ACN) Group to work on the IMOCO4.E project, applying data science techniques to collaborative robotics. His main research interests include machine learning and torque control in robotics.

Juan H. García-Guzmán Juan Helios García Guzmán is a PhD student at the Department of Computer Engineering, Automation and Robotics at University of Granada under the supervision of Niceto Luque and Eduardo Ros. He obtained a B.S in Computer Science in 2021 and a M.S in Data Science in 2022. Within the context of his doctoral studies, his research interests range from the development of infants to robotics learning and cognition.



Nicolás C. Cruz Nicolás Cruz is a post-doctoral researcher at the Department of Computer Engineering, Automation, and Robotics of the University of Granada, Spain. After achieving bachelors and masters degrees in Computer Engineering, he obtained his PhD in Computer Science at the University of Almería, Spain, in 2019. He is a member of the Supercomputing-Algorithms Research Group at that institution. His research focuses on numerical optimization through metaheuristics and high-performance computing applied to different problems, such as the design and control of solar power tower plants, neural model tuning, and optimization of mechanisms.

Brayan Valencia-Vidal Brayan Valencia received his B.Sc. in mechatronics engineering in 2013 from the U. Nacional de Colombia. He is currently working towards the Ph.D. degree at the U. Granada (Spain). He is Assistant Professor and member of the Osiris & Bioaxis Research Group of the U. El Bosque (Colombia). His main research interests include neural networks, dynamic modeling and control of robots.

Niceto R. Luque Dr. Luque, an Associate Professor at the Department of Information and Communication Technologies, Automation, and Robotics (ICAR), earned a Bachelor's in Electronics Engineering and a Master's in Automatics and Industrial Electronics from the University of Cordoba (Spain) in 2003 and 2006, respectively. He later obtained a Ph.D. in Computer Architecture and Networks from the University of Granada in 2013.

His research focuses on computational neuroscience, neurorobotics, and neuromorphic engineering, particularly studying cerebellar adaptation and its role in aging, and applying motor intelligence to cobots via cerebellum simulation.

Eduardo Ros Dr. Eduardo Ros is currently Full Professor at the Department of ICAR (Computer Engineering, Automation and Robotics) at the University of Granada.

He is a very active researcher at an international level, he has participated as IP in 10 European Grants. He leads an interdisciplinary lab, with interest in computational neuroscience, neurorobotics, neuromorphic engineering, real-time image processing, time transfer and synchronization, etc. In particular, his main research interests include simulation of biologically plausible processing schemes with spiking neural networks, neurorobotics, collaborative robotics and time transfer and synchronization techniques, etc.