ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 28, Number 1, 2025, 63–76

Machine Learning Control for Assistive Humanoid Robots Using Blackbox Optimization of PID Loops Through Digital Twins

Andrei MATEESCU[1], Dragos Constantin POPESCU[1,*], Ioana Livia STEFAN[1], Ioana Miruna VLASCEANU[1], Alina Claudia PETRESCU-NITA[2], Ioan SACALA[1], and Ioan DUMITRACHE[1]

[1]Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Splaiul Independentei No. 313, 060042 Bucharest, Romania

[2]Faculty of Applied Sciences, University POLITEHNICA of Bucharest, Splaiul Independentei No. 313, 060042 Bucharest, Romania

Email: andrei.mateescu0612@upb.ro, dragos.popescu@upb.ro*, ioana_livia.stefan@upb.ro, ioana.vlasceanu1512@upb.ro, alina.petrescu@upb.ro, ioan.sacala@upb.ro, ioan.dumitrache@upb.ro

* Corresponding author

Abstract. With the latest technological advancements in Machine Learning, the focus shifted significantly from classical data analysis to controlling various robotics systems. Such an approach, compared to classical control methodologies, provides a simultaneous design, validation and robustness analysis that is performed in an autonomous manner. Although the dynamic performance is not formally guaranteed, the more learning iterations are performed, the more the confidence in the designed solution increases. In this work, we address the problem of accelerating Machine Learning Control (MLC) algorithms by parallelizing the learning with the aid of a simulated test environment containing a Digital Twin of a NAO robot. The increase of the modeling robustness and of the generality of the control algorithm is ensured by performing random positioning tasks with each learning episode. The solutions are further leveraged through Transfer Learning using the real robot and the results are validated and compared. Our main goal is to provide a design framework for assistive robots, which bring significant societal benefits, although require high reliability and safe operation. In this respect, a thorough statistical study concerning the comparison of two typical MLC algorithms, namely the Genetic Algorithm and the Bayesian Optimization, is included.

Key-words: Assistive robots; automation and control theory; digital twins; natural computing; optimization; robotics.

1. Introduction

Machine Learning Control (MLC) is a new paradigm which allows autonomous training, by enabling robots to learn by trial and error in order to maximize an application-specific reward [1]. The learning process can be performed in a simulated environment, in a real one, or in a combination of both [2]. The design of robust control strategies require extensive testing, which in a classical approach is usually performed after the development phase. MLC provides an efficient design framework alternative which performs simultaneously the evaluation of the results, while computing the solution. On using a simulated environment or multiple real ones, further computational improvements can be achieved by performing the learning process in a parallel manner.

The use of Digital Twins (DTs) enables the simulation of real-world robots or systems, so that valuable data can be extracted, such as computing the performance indicators, detecting potential incidents and redundancy, identifying fault tolerance, with the goal of obtaining optimal results on the actual real-world system [3].

The MLC algorithms performed on a physical system has higher costs (in terms of time and resources) compared to using simulated environments [4]. In this way, transfer learning applied in MLC with the aid of DTs is harnessing previously learned knowledge as learning initialization [5]. In our approach, we refer to transfer learning as the use of the solutions learned in simulated robotics environments in its physical counterpart.

Our paper presents the comparison of two MLC methods, a Genetic Algorithm (GA) and a Bayesian Optimization (BO) algorithm, aimed to learn optimal Proportional Integral Derivative (PID) gains for a NAO robot's joints in order to achieve increased positioning performance. The optimization is performed on a DT from a simulated environment, using Gazebo, to transfer the results on the physical robot afterwards. The approach ensures increased modeling robustness and generality of the control algorithm achieved through performing random positioning tasks with each learning episode.

The paper continues in Section 2 with a discussion on the common DT environments and MLC algorithms available in the literature, and an overview of related applications on the topic. Then, the concept and implementation of the algorithms is presented in detail in Section 3, the results of the learning in section 4 and the validation of the solutions on the real robot in Section 5. Finally, conclusions are highlighted in Section 6.

2. Related Work

As ML has recently become ubiquitous when finding solutions across various complex tasks, a consistent effort was put into developing simulation environments and DTs appropriate for specific applications. Among the most popular platforms it is worth mentioning the Gymnasium (formerly OpenAI Gym) which is general-purpose, but with an accent towards developing control algorithms for robotics and dynamical systems, Neural MMO which is used for developing multi-agent systems, Gazebo which is targeted towards developing complex robotics simulation environments with accurate physics and NVIDIA Isaac Gym, which offers a computationally efficient, general purpose robotics simulator which is GPU accelerated.

The field of robotics is one of the main beneficiary of the MLC development. In [6] is developed a MLC strategy for a humanoid THORMANG3 robot balancing on a scooter. The training was performed on NVIDIA Isaac Gym GPU-based simulation environment. The results show

that the controller is capable of successfully balancing the robot-scooter system and tracking the planned trajectories.

For the development of a motion planning algorithm for robotic arms, in [7] is presented a variation based on MLC augmented with data from real arm motions of humans. This approach is built on Deep Deterministic Policy Gradient (DDPG) and Hindsight Experience Replay (HER) algorithms, with a MuJoCo simulated environment to test and improve the learned model before applying it to actual robotic arms. The training and real-world experiments performed show satisfactory results and a success rate of close to 90%.

In [8] is proposed an approach for teaching a NAO robot to perform object placement and grasping tasks by combining a Proximal Policy Optimization (PPO) algorithm with fuzzy logic, used to convert environmental data. The membership function parameters of the fuzzyfication process are further fine-tuned using an Artificial Bee Colony (ABC) optimization algorithm. The simulation environments for training and evaluation were developed in Pybullet.

A hybrid approach between classical control and MLC is presented in [9]. The locomotion algorithm of the NAO robot includes low level actuators control based on a Linear Quadratic Gaussian (LQG) algorithm designed on the mathematical modelling of the robot dynamics. The state estimator is determining the pose of the robot by fusing information from an Inertial Measurement Unit (IMU) sensor and from position sensors of the joints. The control architecture is assisted by a DT while a PPO algorithm is used for modelling the residuals between the robot and the simulated one. This information is fed back to simulate the next time-sample. Finally, a GA is used to fine-tune the trajectory planner parameters.

A method for tuning multi-loop PID controllers using BO for processes described by Single Input Single Output (SISO) models is presented in [10]. Their proposed approach was successfully tested on a non-linear evaporator process simulated in Simulink with the mention that BO has a slower convergence rate for high dimensional problems and to improve it, the optimization problem should be reduced to a lower dimension, breaking the initial problem into several, smaller and easier to optimize ones.

Using BO with Gaussian Process (GP) regression for automatic tuning of PID controllers in aircraft maneuvering, for a path following scenario, is described in [11]. The goal was to find the optimal controller parameters that result in the shortest flight time and the smallest deviation from the destination. The testing was done using a computer aircraft simulator with 6DoF nonlinear dynamics and the results showed that the BO tuned PID controller outperformed the other controllers tested.

In [12], researchers used a nonlinear adaptive PID controller for a mobile manipulator, whose parameters were tuned by BO. The issues of extensive simulations and expensive computations were addressed in [13] by employing a parallel BO algorithm which benefits from a higher efficiency. The approach has been tested on several applications from an industrial bio-oil processing plant and the findings demonstrate the increased efficiency of the parallel BO algorithm when tuning PID controllers in simulated environments.

Research on the topic of optimizing PID controllers using GAs is not new. Various researches were conducted including multi-objective GA for the design of PID controllers [14], the use of GA for PID parameter optimization implemented in Matlab for Linear Time Invariant (LTI) models, which showed that the parameters can be quickly tuned with great performance [15] and Fast GAs, bringing an improvement over regular GAs in terms of convergence [16].

In more recent works, the GAs have been used with satisfactory performance in robotics applications, for mobile robots, aiming at increased performance of speed and accuracy during

navigation [17], for walking control of quadruped soft robots [18] and ladder climbing, where stability and a smooth motion is required [19]. The researchers concluded that GA represents a viable method of PID parameter optimization, rapidly providing results due to a high convergence rate, ultimately enhancing the robots' navigation capabilities.

The improvements in this article over the existing literature include the development of a statistical methodology for evaluating and comparing the performance of two MLC methods, the cost reduction approach through parallel learning using DTs, the transfer learning approach while taking into account the cross-coupling of the joints and increasing the modeling robustness and the generality of the control algorithm by performing random positioning tasks with each learning episode.

3. Methods

The problem of designing a single multi-variable controller for a humanoid robot is not trivial. There is a strong interaction between all the joints due to the mechanical and kinematic configuration of the robot body. The only viable solution is to decouple each control loop and to tune by learning all the PID controllers simultaneously. The approach developed in this paper uses two meta-heuristic algorithms in a MLC setup. Both GA and BO learning algorithms were used in a similar manner for comparison purpose. The aim is to determine a set of optimal gains (namely Kr, Ki and Kd) of the PID controller from every of the 24 joint of a NAO robot, in order to maximize the positioning performance (speed and accuracy).

In the first phase, the learning was performed on a NAO DT in a parallel manner. The resulting solutions are transferred afterwards on a real NAO robot and a thorough statistical comparative analysis of the performance is made. The control loops for the MIMO (multi-input multi-output) dynamical system of the robot body is presented in Fig. 1, where the controller gains are configured according to the values provided by the optimization algorithm.

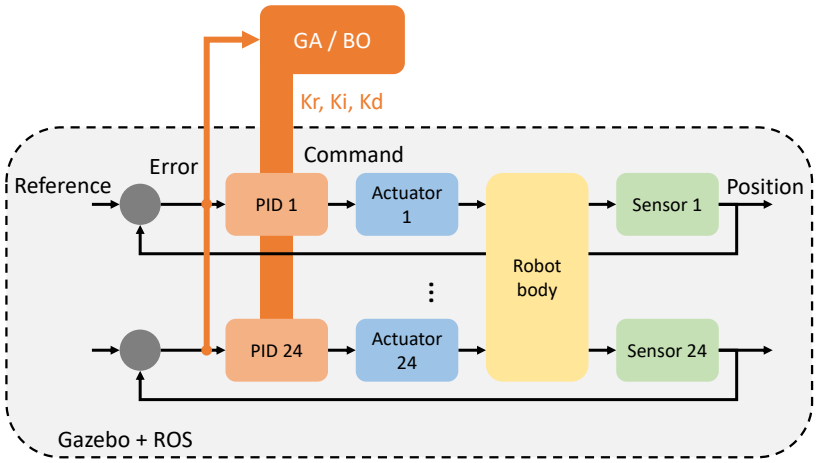
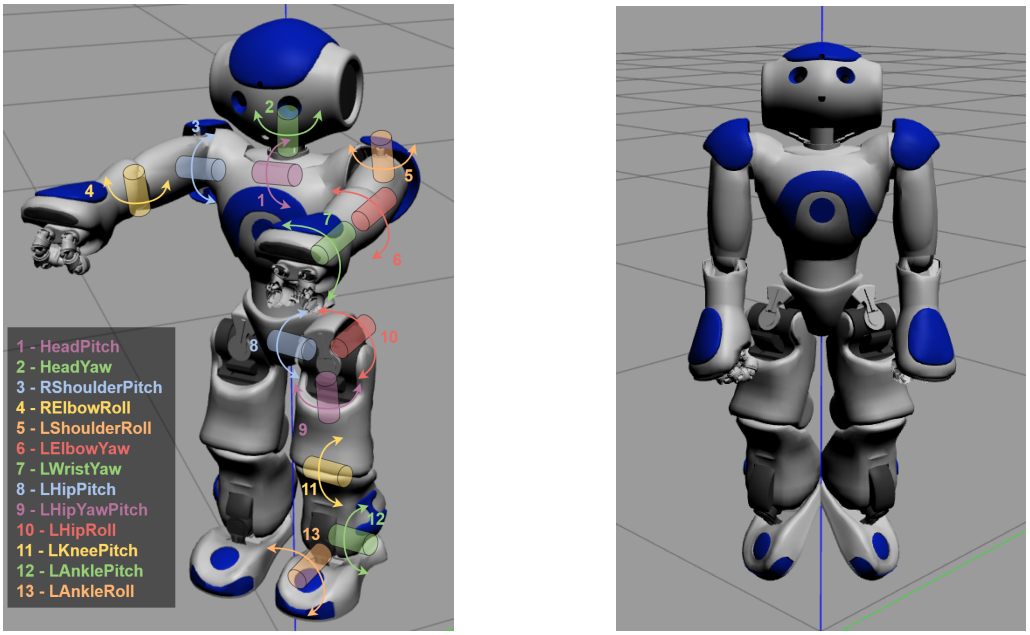


Fig. 1. The control loops of a NAO robot with PID gains tuned by the optimization algorithm.

3.1. Digital Twin setup

The DT was implemented using the Robot Operating System (ROS) and the Gazebo simulation environment. The ROS package was created using the .urdf model from [20]. The simulation ran locally, in parallel, on multiple Virtual Machines, having two 4Ghz GPU cores and 4GB of RAM each. The NAO robot's torso was fixed to the world in order to address only the joint positioning problem. The stability of the robot is ensured at a higher hierarchical level through specific algorithms that provide the reference trajectories for the PID controllers.



(a) (b)

Fig. 2. The naming convention of NAO robot joints (a) and initial pose in simulation (b)

Since the robot is symmetrical, and in order to reduce the complexity of the learning process, only half of the joints are considered. The other half will share the same PID gains. A schematic with all the joints and their naming convention is depicted in Fig. 2a.

Each PID controller was configured for position error input and actuator command output. The mathematical formulation of the PID controller is depicted as follows using the notations $u(t)$ for the actuator command and $\varepsilon(t)$ for the position error.

$$u(t) = K_r\varepsilon(t) + K_i \int_0^t \varepsilon(t)dt + K_d \frac{d\varepsilon(t)}{dt} \tag{1}$$

A configuration file was created for the controllers which were initialized as unity-gain proportional ones. Upon launching the simulation, a Gazebo Empty World is generated, the robot is spawned inside the Gazebo Empty World and the controllers are loaded while awaiting an input position reference. The initial NAO pose is depicted in Fig. 2b.

3.2. Implementation of optimization algorithms

The optimization algorithm and test procedure was implemented in Python, and the interface with the DT simulation environment was achieved using rospy tools. A Python implementation from [21] of an elitist GA was adopted. Each individual is encoding one candidate solution of the 39 PID gains. The BO is implemented in Python as well [22] and the gains are inputs of the black-box function to be optimized.

The gains were trained for all the joint controllers simultaneously, while minimizing a cost defined below. The steps performed during the learning procedure are depicted in Fig. 3.

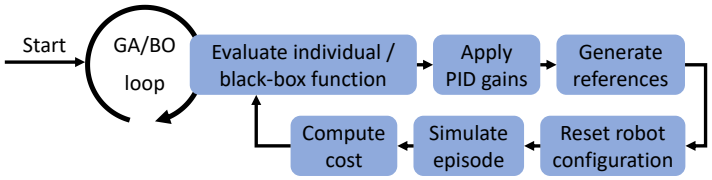


Fig. 3. The steps performed during the optimization procedure.

Considering that each joint has limited travel range, the default position (where the joint is reset) is considered the middle of the interval. For increased generality of the determined solutions and for an increase in modeling robustness, the position reference is chosen randomly at each simulation episode in the $[-max/2; max/2]$ interval (where $[-max, max]$ is the full motion range) in order to accommodate possible overshoots that may occur.

The position of the joints is read with a 100 Hz frequency from the virtual sensors using a subscriber to the *nao/joint_state_controller* topic. The positions are saved and used to compute the cost value for the optimization algorithm as cumulated normalized squared error over all the joints:

$$Cost = \sum_{i=1}^{13} \varepsilon_i \tag{2}$$

where ε_i is the normalized squared error for each joint:

$$\varepsilon_i = \frac{\sum_t [r_i - y_i(t)]^2}{r_i - r_{i_0}} \tag{3}$$

and r_i is the reference position of joint i, r_{i_0} its default position and $y_i(t)$ the evolution of the position during the motion.

The error is normalized by the amplitude of the reference step in order to make the contribution of all the joints to the cumulative cost uniform. Minimizing the square error of the position is ensuring fast transient regimes with small overshoots. The cost is fed back to the optimization algorithm and used for computing the PID parameters of the next learning generation.

4. Optimization Results

For the GA, in order to optimize its hyper-parameters, a number of 12 initial experiments were performed. The results are summarized in Table 1. For a similar population size of 50

individuals, over 20 generations and an uniform crossover, the values of the Mutation probability (Mut. prob.), Elite ratio, Crossover probability (Cross. prob.) and Parents portion (P. p.) were varied. For each hyper-parameters choice, the learning was ran three times in order to increase the confidence in the results. The cumulative cost results for each of the three trials and their average is highlighted (expressed in rad^2).

Table 1. Experiments to find optimal GA hyper-parameters

Exp. no.	**Mut. prob.**	**Elite ratio**	**Cross. prob.**	**P. p.**	**#1 cost**	**#2 cost**	**#3 cost**	**Av. cost**
1	0.1	0	0.3	0.1	1371.72	1012.08	1109.69	1164.50
2	0.3	0	0.3	0.1	1112.50	450.82	511.69	691.67
3	0.5	0	0.3	0.1	794.11	1098.99	1453.36	1115.49
5	0.3	0.02	0.3	0.1	1602.42	832.44	937.57	1124.14
6	0.3	0.1	0.3	0.1	619.25	441.07	331.57	463.96
8	0.3	0.1	0.5	0.1	171.50	736.42	636.31	514.74
9	0.3	0.1	0.7	0.1	522.14	518.51	531.83	524.16
11	0.3	0.1	0.5	0.3	350.89	593.28	591.37	511.85
12	0.3	0.1	0.5	0.5	1014.55	839.49	538.72	797.59

The total running time was approximately 1.5 h. The experiments show that the best results were achieved using a Mutation probability of 0.3, an Elite ratio of 0.1, a Crossover probability of 0.5 and a Parents portion of 0.2.

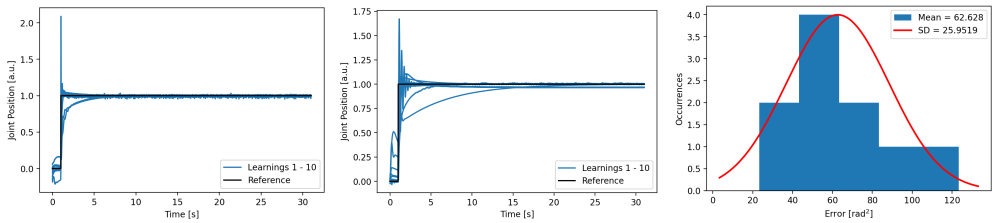
Having established optimal values for these hyper-parameters, a consequent set of two experiments were performed to analyze how the running time is scaling with the computational complexity brought by the population size and the number of generations. Again, the experiments were performed three times for an increased confidence in the results, which are summarized in Table 2 and where the costs are also expressed in rad^2.

Table 2. GA experiments with different number of generations and population size

Exp. no.	**No. of Gens.**	**Pop. Size**	**Time**	**#1 cost**	**#2 cost**	**#3 cost**	**Av. cost**
1	50	200	$\sim$ 15.5h	114.32	119.86	24.51	86.23
2	200	50	$\sim$ 15h	247.76	195.53	313.98	252.42

The results clearly show that the contribution of the Population size is similar with the one of the Number of generations and that the time linearly scales with the number of times the cost is evaluated. Consequently, for the GA learning, a population of 250 individuals with 60 generations were selected with a simulation time of approximately 23 h.

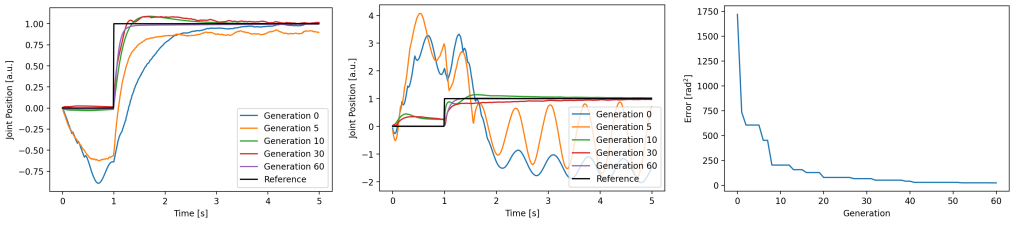
The learning process with the GA was performed 10 times, with different random initialization, in order to perform a statistical analysis and to built trust on the results. In Fig. 4a and Fig. 4b two of the 13 joints were picked (LKneePitch and LAnklePitch respectively) and a step response for the best PID gains achieved after the ten learnings are presented. Considering that the step amplitude is randomly picked, for comparison purpose, the responses were scaled to $[0, 1]$ interval and therefore, the values are expressed as arbitrary units. In Fig. 4c the histogram of the residual cumulative cost (the cost of the best generation) of the ten learnings is depicted, showing an approximate normal distribution with a mean value of 62.63 rad^2.



(a) (b) (c)

Fig. 4. Step response for two picked joints (LKneePitch - 4a, LAnklePitch - 4b) using the solutions from the GA learnings and the histogram of their residual cumulative cost (4c).

For the same two joints, in Fig. 5a and 5b the improvement of performance over the GA generations of one of the ten learnings is highlighted. The step responses of the initial, fifth, tenth, thirtieth and sixtieth generation are again scaled for comparison purpose. There is a clear improvement in performance which is also emphasized in Fig. 5c where the evolution of the cumulative cost is depicted.



(a) (b) (c)

Fig. 5. The improvement of performance during one of the ten GA learnings for LKneePitch (5a) and LAnklePitch (5b) joints and the evolution of the cumulative cost (5c).

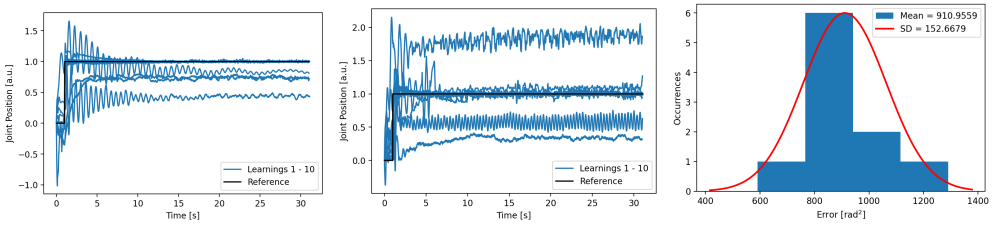
For the BO, in order to pick its hyper-parameters, namely the number of initial random evaluation points and the number of subsequent iterations, a set of 16 experiments were performed in order to assess the correlation between the computational complexity, the execution time and the performance. Similarly to the GA, each experiment was performed three times in order to increase the confidence in the results. The conclusions are summarized in Table 3 which show that the more initial evaluations (Init. eval.) and subsequent iterations (No. iter.), the better the performance (quantified as the cumulative cost and expressed in rad^2), however, the latter having a stronger impact in the simulation time due to added computational complexity of updating the GP (Gaussian Process) model with an increasingly number of evaluation points.

The learning process with the BO was performed 10 times, with a chosen number of initial evaluations of 400 and subsequent iterations of 600 with a simulation time of approximately 22 h. Similar with the GA case, in Fig. 6a and Fig. 6b are depicted the step responses of the LKneePitch and LAnklePitch joints with the solutions of the ten learnings, while in Fig. 6c is presented the histogram of their residual cumulative cost showing an approximate normal distribution with a mean value of 910.95 rad^2.

The improvements of performance over the BO iterations for one of the learnings is presented

Table 3. BO experiments for choosing its hyper-parameters

Exp. no.	Init. eval.	No. iter.	Time	#1 cost	#2 cost	#3 cost	Av. cost
1	10	10	∼ 3min	1739.55	1618.77	2172.00	1843.44
2	10	100	∼ 16min	1691.82	1723.98	1578.11	1664.60
3	10	200	∼ 46min	987.27	1095.07	1680.43	1254.26
4	100	10	∼ 13min	1503.28	2005.56	2434.90	1981.24
5	100	100	∼ 38min	1336.35	1490.86	2150.73	1659.31
6	100	200	∼ 1.3h	927.56	1085.61	786.79	933.32
7	200	10	∼ 25min	1000.53	1326.03	1345.73	1224.10
8	200	100	∼ 1h	1649.09	1161.95	1241.74	1350.93
9	200	200	∼ 1.8h	663.54	1354.79	1124.41	1047.58
10	100	400	∼ 3.5h	979.07	1037.78	1191.87	1069.58
11	200	400	∼ 5h	813.81	799.33	932.34	848.49
12	400	400	∼ 9h	935.72	1173.26	385.01	831.33
13	400	100	∼ 2h	679.88	1428.89	1086.55	1065.11
14	400	200	∼ 4h	1297.42	1043.57	894.50	1078.50
15	400	600	∼ 22h	597.86	717.64	733.99	683.16
16	600	400	∼ 19h	682.18	703.13	856.10	747.14



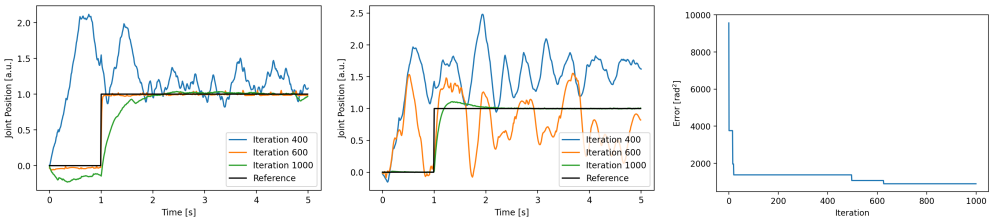
(a) (b) (c)

Fig. 6. Step response for two picked joints (LKneePitch - 6a, LAnklePitch - 6b) using the solutions from the BO learnings and the histogram of their residual cumulative cost (6c).

in Fig. 5a for LKneePitch and in Fig. 5b for LAnklePitch. It is worth noting that the relevant analysis is for the subsequent BO iterations because the random evaluations are only an initialization of the algorithm. Considering the step responses of the 400th, 600th and 1000th iterations, there is a clear improvement in performance confirmed by the evolution of the cumulative cost depicted in Fig. 7c.

Although there is a clear improvement in performance during both GA and BO learning, there is also a clear difference between the results. Comparing the step responses from Fig. 4a and Fig. 4b with the ones in Fig. 6a and Fig. 6b, it can be highlighted that the BO PID gains achieve higher instability with joint oscillations in the steady state and nonzero tracking error. The difference in performance is confirmed as well by the histograms of residual cumulative cost (Fig. 4c and Fig. 6c) showing a difference of an order of magnitude in the favor of the GA.

The learning duration was approximately 23 h for both the GA and BO, but the difference in performance may be due to the fact that during each GA learning, 15.000 DT simulations



(a) (b) (c)

Fig. 7. The improvement of performance during one of the ten BO learnings for LKneePitch (7a) and LAnklePitch (7b) joints and the evolution of the cumulative cost (7c).

were performed and only 1.000 in the BO case. This brings to the conclusion that in terms of complexity (duration vs. number of evaluations), the GA is more efficient, while in terms of performance (number of evaluations vs. performance) the BO is more efficient.

Additionally, with the adoption of a cumulative cost, the performance is not uniform along all of the joints. Although most of the controllers may have a very good performance, there may be some which perform poorly, having a detrimental impact to the cost. This issue can be solved by adopting multi-objective GA and BO variants, where the performance is simultaneously optimized for each individual joint.

5. Validation on the Real Robot

Considering that the BO results were not satisfactory, only the solutions of the ten GA learnings performed using the DT were transfered to a real NAO robot for validating and comparing the results. Having performed all due diligence and to the best of our knowledge, at the time of writing this article, the NAO robot doesn't provide any means of configuring the PID gains of the joints. Consequently, the performance comparison is concerning the results achieved within the DT simulation and with the real NAO robot (having default PID gains) in two distinct scenarios, when a step reference is provided and when a reference trajectory made of position points from the simulated response is considered, in order to force the real response to mimic the simulated one, as in a cascade loop architecture.

The comparison is performed on the LShoulderRoll joint whose movement is easily observable and qualitatively analyzed (as depicted in Fig. 8). The motion is covering the entire travel range for the same purpose.

In Fig. 9 are depicted the responses of the ten sets of PID gains learned using the GA: of the DT simulated NAO (Fig. 9a), of the real NAO robot to the step reference (9b) and to the reference trajectory (9c).

With a simple qualitative analysis of the plots, it can be concluded that the simulated responses achieve the best performance with fast settling times, no overshoot and tracking errors. On the real NAO robot, there is a moderate performance degradation with increased settling time, very small overshoot and a noticeable tracking error. The tracking error is improved using the reference trajectory, however with an increased overshoot and a small settling time degradation. Overall, in practical positioning applications, the tracking error is of a higher importance against the emerging of small overshoot and settling time degradation.

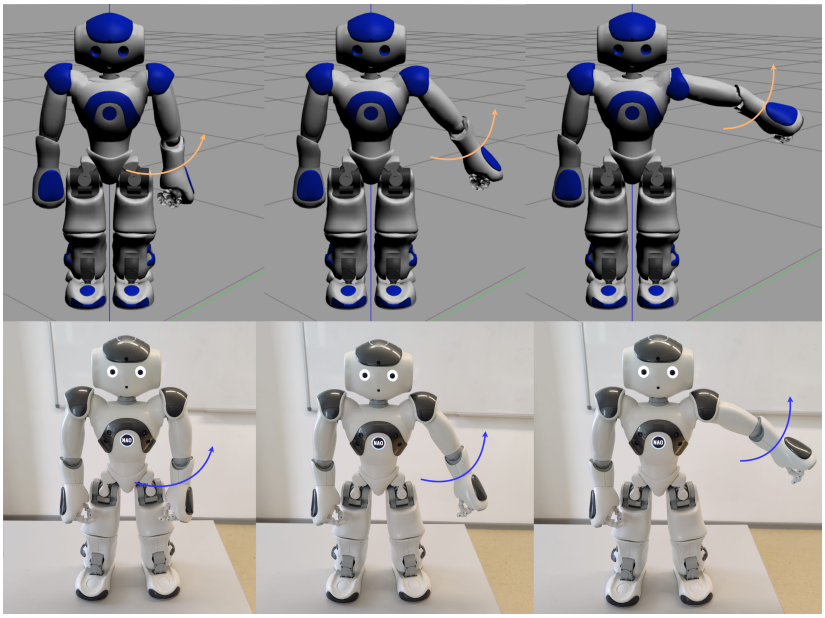
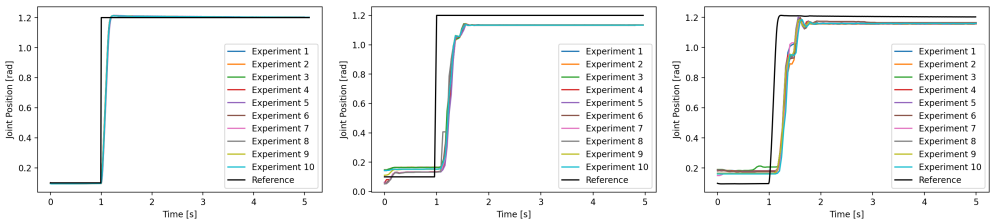


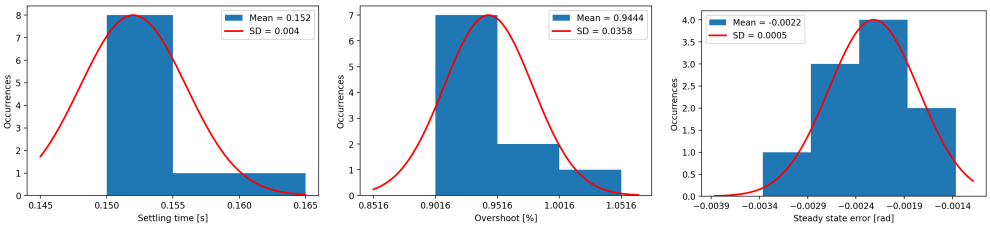
Fig. 8. DT simulated robot (upper half) versus real NAO robot (lower half) while performing a LShoulderRoll movement.



(a) (b) (c)

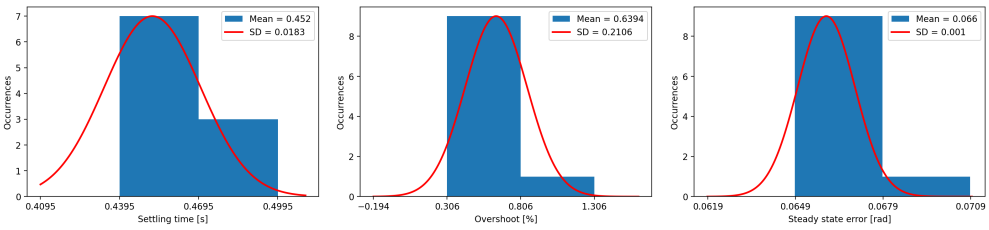
Fig. 9. Position evolution for LShoulderRoll joint in the DT simulated environment (9a), on the real NAO with step reference (9b) and with the reference trajectory (9c) when using the ten sets of PID gains learned using the GA.

To further compare the results, a quantitative statistical analysis was performed. For each of the ten solutions within all the three test scenarios, standard dynamical performance metrics (namely the settling time, the overshoot and the steady state error) were determined and plotted on histograms. In Fig. 10 are depicted the histograms for the DT simulated NAO robot, in Fig. 11, the step response on the real NAO robot and in Fig. 12, the reference trajectory response on the real NAO robot. The histograms show an approximate normal distribution in most of the cases and their mean value confirm the conclusions drawn above. With the DT simulated NAO robot, the PID gain solutions provide the best performance: a fast settling time with a mean value of 0.15 seconds, an almost non-existent overshoot with a mean value of 0.94% and a close to zero steady state error. The real NAO robot in both of the test scenarios achieve similar settling



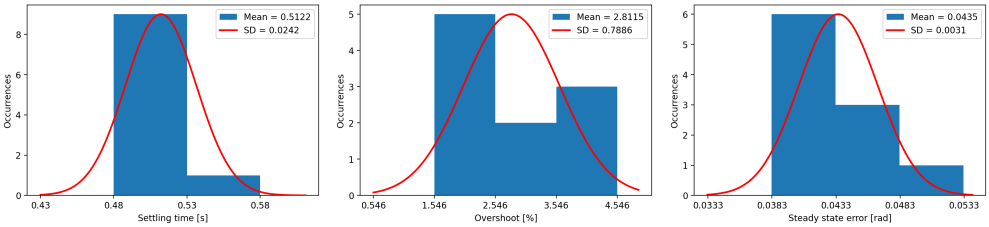
(a) (b) (c)

Fig. 10. Settling time (10a), overshoot (10b) and steady state error (10c) histograms for the step responses of the DT simulated NAO robot.



(a) (b) (c)

Fig. 11. Settling time (11a), overshoot (11b) and steady state error (11c) histograms for the step responses of the real NAO robot.



(a) (b) (c)

Fig. 12. Settling time (12a), overshoot (12b) and steady state error (12c) histograms for the reference trajectory responses of the real NAO robot.

times with mean values of around 0.5 seconds. The step response achieve a small overshoot with a mean value of 0.63%, however, with a noticeable steady state error of around 0.06 radian. The latter is decreased with 35% on average using the reference trajectory, while the overshoot increases to a mean value of 2.8%.

6. Conclusions

The control of real dynamical systems, even with low structural complexity and in simple environments, can prove difficult to design. The lack of simple mathematical models that cap-

ture precisely enough the interaction between the system and its environment makes classical gradient-based optimization algorithms unpractical.

With the advancement of the MLC algorithms integrated with DTs embedding simple physics models, a new approach to design and to evaluate in real-time control algorithms emerged. As the computational power become easily available, the obvious benefit over classical approaches is the parallelization capabilities. Tens to thousands of simulation instances can be executed at the same time with minimum marginal cost, while the learning is significantly accelerated. An additional benefit is the merger of the algorithm design and validation phases. The stability and the dynamical performance achieved by the iteratively refined solution has a high degree of trust statistically proven. Through transfer learning, the solutions are then implemented on the real dynamical system and can be further refined to encompass the modeling uncertainties and simulation residuals.

In this paper, we addressed the problem of systematically and statistically comparing two typical optimization algorithms in a DT environment, a GA and a BO, and serves as a framework for harnessing the solutions through transfer learning on real robotic systems. The use of the NAO robot was only to illustrate the capabilities of this approach considering that it is a complex mechanical and dynamical system. However, the method can be applied for any dynamical system to which a DT is available.

Our findings show that there is a clear difference in performance and efficiency between the GA and BO. The BO is more computationally intensive achieving fewer simulation episodes over the same amount of time compared to the AG. This makes the overall performance with an order of magnitude worse. However, scaling the performance by the number of simulation episodes, the BO seem to perform better and to understand more the problem it needs to optimize.

Testing the GA solution (which has better performance) on the real robot show that the robustness is good with moderate performance degradation compared to the simulation case. This result was expected considering that the solution was not further refined due to the software constraints of the NAO robot discussed in the previous section. Future work will include the use of different mobile and humanoid robots with more flexible software features.

Acknowledgement. This work was supported by the National Program for Research of the National Association of Technical Universities - GNaC ARUT 2023, grant number 83/11.10.2023 (optDDNc).

References

[1] J. M. D. DELGADO and L. OYEDELE, *Robotics in construction: A critical review of the reinforcement learning and imitation learning paradigms*, Advanced Engineering Informatics **54**, 2022, paper 101787.

[2] E. KAUFMANN, L. BAUERSFELD, A. LOQUERCIO, M. MÜLLER, V. KOLTUN and D. SCARAMUZZA, *Champion-level drone racing using deep reinforcement learning*, Nature **620**, 2023, pp. 982–987.

[3] M. MATULIS and C. HARVEY, *A robot arm digital twin utilising reinforcement learning*, Computers & Graphics **95**, 2021, pp. 106–114.

[4] K. KADAMALA, D. CHAMBERS and E. BARRETT, *Enhancing HVAC control systems through transfer learning with deep reinforcement learning agents*, Smart Energy **13**, 2024, paper 100131.

[5] F. SHOELEH and M. ASADPOUR, *Graph based skill acquisition and transfer learning for continuous reinforcement learning domains*, Pattern Recognition Letters **87**, 2017, pp. 104–116.

[6] J. BALTES, G. CHRISTMANN and S. SAEEDVAND, *A deep reinforcement learning algorithm to control a two-wheeled scooter with a humanoid robot*, Engineering Applications of Artificial Intelligence **126**, 2023, paper 106941.

[7] A. YANG, Y. CHEN, W. NAEEM, M. FEI and L. CHEN, *Humanoid motion planning of robotic arm based on human arm action feature and reinforcement learning*, Mechatronics **78**, 2021, paper 102630.

[8] P.-H. KUO and K.-L. CHEN, *Two-stage fuzzy object grasping controller for a humanoid robot with proximal policy optimization*, Engineering Applications of Artificial Intelligence **125**, 2023, paper 106694.

[9] M. KASAEI, M. ABREU, N. LAU, A. PEREIRA and L. P. REIS, *Robust biped locomotion using deep reinforcement learning on top of an analytical control approach*, Robotics and Autonomous Systems **146**, 2021, paper 103900.

[10] J. COUTINHO, L. SANTOS and M. REIS, *Bayesian optimization for automatic tuning of digital multi-loop PID controllers*, Computers & Chemical Engineering **173**, 2023, paper 108211.

[11] D. KIM and H.-S. OH, *Black-box optimization of PID Controllers for aircraft maneuvering control*, International Journal of Control Automation and Systems **20.3**, 2022, pp. 703–714.

[12] H. HAJIEGHRARY, M. P. DEISENROTH and Y. BEKIROGLU, *Bayesian optimization-based nonlinear adaptive PID controller design for robust mobile manipulation*, Proceedings of 2022 IEEE 18th International Conference on Automation Science and Engineering, Mexico City, Mexico, 2022, pp. 1009–1016.

[13] Q. HE, Q. LIU, Y. LIANG, W. LYU, D. HUANG and C. SHANG, *Risk-averse PID tuning based on scenario programming and parallel Bayesian optimization*, Industrial & Engineering Chemistry Research **64.1**, 2024, pp. 564–574.

[14] A. HERREROS, E. BAEYENS and J.R. PERÁN, *Design of PID controllers using multiobjective genetic algorithms*, IFAC Digital Control: Past, Present and Future of PID Control, 2000, pp. 277–282.

[15] Q.F. TENG, D.W. FAN and J. CAO, *Optimized PID controller via genetic algorithm*, Proceedings of 7th International Conference on Electronic Measurement and Instruments, Beijing, China, 2005, vol. 5, pp. 15–19.

[16] X. MENG and B. SONG, *Fast genetic algorithms used for PID parameter optimization*, Proceedings of 2007 IEEE International Conference on Automation and Logistics, Jinan, China, 2007, pp. 2144–2149.

[17] H. GOUD, P.C. SHARMA, K. NISAR, M. R. HAQUE, A. A. A. IBRAHIM, N. S. YADAV, P. SWARNKAR, M. GUPTA and L. CHAND, *Metaheuristics algorithm for tuning of PID controller of mobile robot system*, CMC - Computers Materials & Continua **72.2**, 2022, pp. 3481–3492.

[18] H. MENG, S. ZHANG, W. ZHANG and Y. REN, *Optimizing actual PID control for walking quadruped soft robots using genetic algorithms*, Scientific Reports **14.1**, 2024, paper 25946.

[19] D. SHAH, J. DAVE, D. DAVE, S. SONI and K. PATEL, *Design and development of a quadruped ladder climbing robot using optimized PID parameters using genetic algorithm*, Proceedings of 2024 International Journal of Interactive Design and Manufacturing, 2024, DOI: 10.1007/s12008-024-02167-5.

[20] *Ros-naoqi, nao robot repository*. Accessed: Oct. 7, 2024. [Online]. Available: https://github.com/ros-naoqi/nao_robot.

[21] R. SOLGI, *geneticalgorithm*. Accessed: Oct. 7, 2024. [Online]. Available: https://github.com/rmsolgi/geneticalgorithm.

[22] J. JIMÉNEZ-LUNA and J. GINEBRA, *pygpgo: Bayesian optimization for Python*, The Journal of Open Source Software **2**, 2017, paper 431.