A Software Ecosystem for Research in Learning-Based Receding Horizon Control

Andrea Patrizi*†, Carlo Rizzardo* and Nikos G. Tsagarakis*

***Abstract*— Robotics research in manipulation and locomotion is undergoing a transformative shift towards the use AI-based tools, where purely learning-based control policies and pipelines are starting to take over the field. Despite being shown to be capable of remarkable robustness and performance even when applied to real-world environments, some key inherent limitations such as safety guarantees, interpretability, sample efficiency, persist. For this reason, it is the authors' belief that more classical control approaches should not be considered outdated yet. We thus advocate for a hybrid approach, combining offline data-based policy design, specifically through Reinforcement Learning (RL), with classical online Motion Planning, i.e. Receding Horizon Control (RHC). Even though this kind of hybrid approaches are not entirely new, to the authors' knowledge, there is no specific tool currently available for search in this domain. To this purpose, we developed a modular software ecosystem, hereby briefly presented in its main components and features. To facilitate its usability and diffusion, we made all the core components open source under the GPLv2 license. Furthermore, to showcase the potential of our framework, we briefly present a proof-of-concept example combining a high-level RL agent coupled with a lower-level MPC controller for the execution of a simple locomotion task on a simulated quadruped robot.**

I. A Hystorical Perspective: *from Markov Decision Processes and Dynamic Programming to modern Receding Horizon Control and Reinforcement Learning*

State-of-the-art of locomotion and manipulation pipelines have been shown to be capable of remarkable performance and robustness [1]–[4]. These results stand on the shoulders of more than seventy years of research in robotics, control and learning, starting from the very first industrial automated robot *Unimate* in the 1950s [5], Richard Bellman's pioneering work in the late 1950s and early 1960s on **Markov Decision Processes** (MDPs) [6] and **Dynamic Programming** [7] (DP) and the birth of **Artificial Intelligence** (AI) as an established field of study thanks to the contributions of researchers like Alan Turing, John McCarthy, Marvin Minsky (*connectionism*) and Claude Shannon (*information theory*). MDPs are a mathematical framework used to model sequential decision-making in situations where outcomes are influenced by potentially probabilistic transitions and the actions taken by a decision-maker. MDPs are described by a set of state S, representing the possible configurations or conditions of the system being modeled. At any given time, the system is in one of these states. For each state in the MDP, there is a set of possible actions A that the decision-maker can choose from. Upon taking an action $a \in A$ in a particular state $s \in S$, the system transitions to a new state $s' \in S$, with associated immediate reward $r(s,a,s')$, according to a probability distribution $p(s' \mid s,a)$. The objective in MDPs is to find an optimal policy $\pi^*(a \mid s)$ that maximizes the cumulative (discounted) reward $R_t = \sum_{k=0}^{n-1} \gamma^k \cdot r_{t+k}$ obtained by the decision-maker over a time horizon, possibly infinite. In the late 1950s, Richard Bellman and other researchers introduced the so-called **Bellman equation** for the *value function* $V^\pi(s) = \mathbb{E}_\pi[R_t | S_t = s]$:

$$V^*(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma V^*(s') \right] \quad (1)$$

which, in conjunction with the formulation of MDPs laid the foundations of DP as a systematic method for solving sequential decision-making problems by breaking them down into simpler sub-problems [7]. This resulted in the so-called *Value Iteration* algorithm, which allows to solve for the optimal value function by recursive application of equation (1) and consequently retrieve the optimal policy. Short after, the *Policy Iteration* algorithm [8] was proposed as a way to compute the optimal policy by alternating between value function evaluation and policy improvement. Later in the 1980s, *TD-learning* was introduced [9], with the idea of bootstrapping the learning of value functions for MDPs by using the current value estimate of temporally successive steps, allowing the development of more computationally efficient learning algorithms. In 1889 Watkins introduced *Q-learning*, which combines TD-learning with DP [10] in an off-policy (any policy can be used for experience generation) value-based algorithm. Differently from previous approaches, it tries to estimate a "quality function" Q(s, a), which indicates the value of being is state s and choosing action a. The increased popularization of back-propagation [11] as a way of training powerful function approximators based on neural networks and the ever-increasing computational resources progressively paved the way to the ancestors of today's most successful and employed on and off-policy RL algorithms PPO [12] and SAC [13], respectively. Some of these ancestors include, for instance, the *Natural Policy Gradient* [14], *Deep Quality Networks* (DQN) [15], *Deep Deterministic Policy Gradient* (DDPG) [16] and *Trust Region Policy Optimization* [17] algorithms.

In an analogous way, Dynamic Programming (DP) principles served as a foundational basis for the evolution of modern RHC [18]. DP's emphasis on backward induction and the Bellman equation, which form the cornerstone of its problem-solving methodology, inspired the development of RHC's iterative optimization strategy. Just as DP breaks down complex problems into smaller subproblems

† Department of Informatics, Bioengineering, Robotics and Systems Engineering, Università di Genova, Via All'Opera Pia 13, 16145 Genova.

* Humanoids and Human-Centred Mechatronics (HHCM), Istituto Italiano di Tecnologia (IIT), Via San Quirico 19d, 16163 Genova.

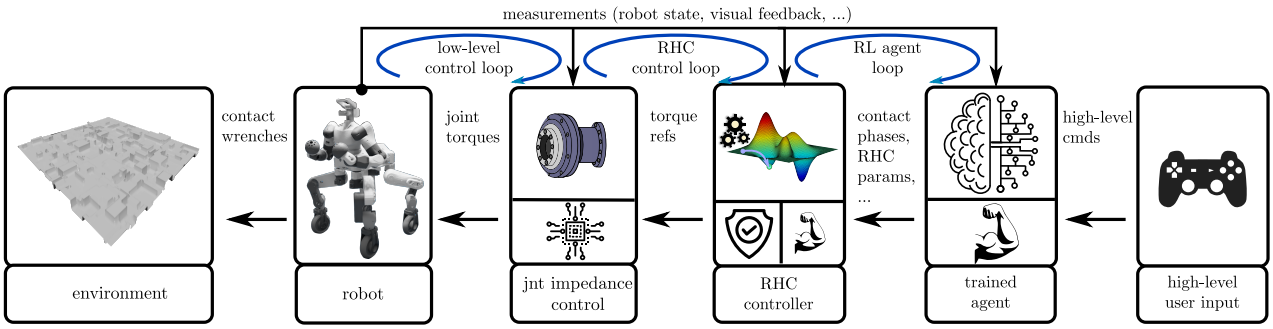

Fig. 1. Our take on Learning-based Receding Horizon Control: a MPC controller is exposed to a RL agent through key runtime parameters, like contact phases, its internal state (costs, constrains..) and interfaces for setting task commands. The agent learns to exploit the underlying RHC controller to perform the tracking of user-specified high-level task references. This allows to both tackle problems which are non-trivial at the MPC level (like phase selection), while also exploiting the flexibility of the agent to complete tasks and the capability of the MPC of ensuring safety.

and iteratively finds optimal solutions by considering future consequences, RHC iteratively solves finite-horizon optimal control problems over shorter time intervals, considering system dynamics and constraints.

Over the years, many algorithms for the solution of the classical receding-horizon nonlinear optimization program

$$\begin{aligned} \underset{u_0,\dots,u_{N-2};x_1,\dots,x_{N-1}}{\text{minimize}} \quad & L(x,u) \\ \text{subject to} \quad & x_{k+1} = f(x_k,u_k),\ \forall k \in [0,N-2] \\ & x_k \in \mathscr{X},\ \forall k \in [0,N-1] \\ & u_k \in \mathscr{U},\ \forall k \in [0,N-2] \\ & x_0 = x_{\text{init}} \end{aligned} \tag{2}$$

have been developed, and several of them have direct ties with DP and, specifically, *Differential Dynamic Programming* (DDP) [19]–[22].

II. A HYBRID APPROACH: *Learning-Based Receding Horizon Control with Reinforcement Learning*

Most of the currently employed control tools rely either on online "model-based" controllers [3], [23] (e.g. Receding Horizon Control) or "model-free" learned policies (e.g. Reinforcement Learning) [1], [2], [24]–[36], which are usually trained offline. Both approaches are historically deeply rooted in DP and MDPs. In the past years there have been several attempts at combining-learning based method and receding horizon controllers [37], [38]. Specifically, the following main approaches can be identified [39]:

1) *Model augmentation*: integration of learned models into RHC controllers to improve prediction accuracy and control performance [24]–[26].
2) *Adaptive tuning* and *parameter optimization*: RHC parameters tuning (e.g. weights, costs, constraints), based on real-time data [27]–[31].
3) *Safety*: a learned-policy is coupled with a RHC controller, which in this context takes the role of a *safety filter* [32]–[36].

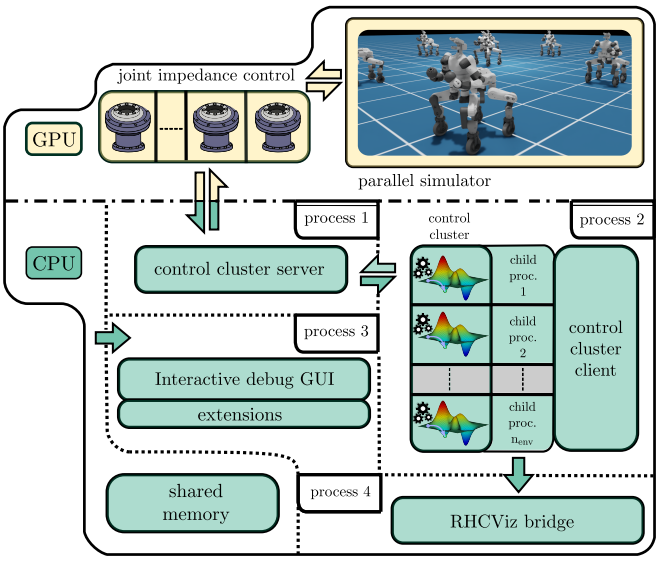

Fig. 2. High-level overview of the software implementation of the training environment to which the agent is exposed: the robot in the simulator is controlled through a joint-level impedance controller, which is in turn used by a higher-level receding horizon controller. The agent can indirectly control the robot through the latter.

III. IMPLEMENTATION: *available tools, framework overview and rationale*

Trajectory optimization,
parallel simulation (isaac, Mujoco)
neural networks
PPO/SAC

[41] [42]

Fig. 2 shown a high level overview if the main moduli the echosystem is composed of. Specifically, it makes use of the following software packages:

- Omniverse *IsaacSim* [40] is employed for performing the parallel GPU-accelerated simulation.
- *Horizon* [43] is employed for formulating the RHC controller, with an iLQR solver employed for the actual solution. Each controller is spawned in its own process using python's multiprocessing module.

- *SharsorIPCpp* [48] is employed as a shared memory backend for fast data sharing between all the components on CPU.
- *OmniRoboGym* [45] is used as a wrapper around Isaac-Sim and provides the *simulation* environment.
- *CoClusterBridge* [46] is in charge of handling the connection and synchronization between the simulation environment and a cluster of RHC controllers. It furthermore provides an extensible debug GUI for monitoring the cluster and abstractions for the controllers.
- *RHCViz* [47] is a debug tool based on ROS1/ROS2 and RViz for visualizing RHC solutions in real-time. For our specific use case, it also allows to inspect a single environment during training without the need of any rendering on the simulator side.
- *LRHControl* [44] is the main package of the echosystem and is responsible for setting up the simulation environment, the control cluster, the (remote) training environment, the debug GUI (extensions included) and it currently utilizes a custom implementation of PPO [12] for training agents.

IV. A PROOF-OF-CONCEPT EXAMPLE: *learning acyclic stepping for locomotion*

REFERENCES

[1] L. Schneider, J. Frey, T. Miki, and M. Hutter, "Learning risk-aware quadrupedal locomotion using distributional reinforcement learning," *arXiv preprint arXiv:2309.14246*, 2023.

[2] T. Miki, J. Lee, L. Wellhausen, and M. Hutter, "Learning to walk in confined spaces using 3d representation," 2024.

[3] B. Dynamics, "Atlas Gets a Grip." `https://www.youtube.com/watch?v=-e1_QhJ1EhQ`, 2023. [Online; accessed 27-March.-2024].

[4] B. Dynamics, "Stepping Up, Reinforcement Learning with Spot." `https://www.youtube.com/watch?v=Kf9WDqYKYQQ`, 2023. [Online; accessed 23-March.-2024].

[5] M. Xu, J. M. David, S. H. Kim, *et al.*, "The fourth industrial revolution: Opportunities and challenges," *International journal of financial research*, vol. 9, no. 2, pp. 90–95, 2018.

[6] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[7] R. Bellman and R. Kalaba, "Dynamic programming and adaptive processes: mathematical foundation," *IRE Transactions on Automatic Control*, no. 1, pp. 5–10, 1960.

[8] R. A. Howard, "Dynamic programming and markov processes," 1960.

[9] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[10] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[14] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[18] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, 2023.

[19] D. H. Jacobson and D. Q. Mayne, "Differential dynamic programming," *(No Title)*, 1970.

[20] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 300–306, IEEE, 2005.

[21] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," *Nonlinear model predictive control: towards new challenging applications*, pp. 391–417, 2009.

[22] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, IEEE, 2012.

[23] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[24] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," 2012.

[25] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning multi-step prediction models for receding horizon control," in *2018 European Control Conference (ECC)*, pp. 1335–1340, IEEE, 2018.

[26] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer, "Learning-based robust model predictive control with state-dependent uncertainty," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 442–447, 2018.

[27] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 491–496, IEEE, 2016.

[28] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 270–277, IEEE, 2016.

[29] F. D. Brunner, M. Lazar, and F. Allgöwer, "Stabilizing model predictive control: On the enlargement of the terminal set," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 15, pp. 2646–2670, 2015.

[30] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

[31] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[32] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE conference on decision and control (CDC)*, pp. 6059–6066, IEEE, 2018.

[33] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[34] J. H. Gillulay and C. J. Tomlin, "Guaranteed safe online learning of a bounded system," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2979–2984, IEEE, 2011.

[35] K. P. Wabersich and M. N. Zeilinger, "Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning," *arXiv preprint arXiv:1812.05506*, 2018.

[36] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[37] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[38] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5973–5979, IEEE, 2021.

[39] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.
[40] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.
[41] "Mujoco 3."
[42] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.
[43] F. Ruscelli, A. Laurenzi, N. G. Tsagarakis, and E. M. Hoffman, "Horizon: a trajectory optimization framework for robotic systems," *Frontiers in Robotics and AI*, vol. 9, 2022.
[44] A. Patrizi, "LRHControl." `https://github.com/AndrePatri/LRHControl`, 2023. [Online; accessed 10-March.-2024].
[45] A. Patrizi, "OmniRoboGym." `https://github.com/AndrePatri/OmniRoboGym`, 2023. [Online; accessed 10-March.-2024].
[46] A. Patrizi, "CoClusterBridge." `https://github.com/AndrePatri/CoClusterBridge`, 2023. [Online; accessed 10-March.-2024].
[47] A. Patrizi, "RHCViz." `https://github.com/AndrePatri/RHCViz`, 2023. [Online; accessed 10-March.-2024].
[48] A. Patrizi, "SharsorIPCpp." `https://github.com/AndrePatri/SharsorIPCpp`, 2023. [Online; accessed 10-March.-2024].