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, etc.. 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 For this reason, it is the authors' belief that more classical control approaches should not be considered outdated yet.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 synthetically 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** [7] (MDPs) and **Dynamic Programming** [8] 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*).

Most of the currently employed control tools rely either on online "model-based" controllers [3], [6] (e.g. Receding Horizon Control) or "model-free" learned policies (e.g. Reinforcement Learning) [1], [2], which are usually trained offline. In the past there have also been several attempts at combining both [], where these main trends emerge:

1) *Model augmentation*: integration of learned models into the RHC controller to improve prediction accuracy and control performance.
2) *Adaptive tuning* and *parameter optimization*: RHC parameters tuning, based on real-time data.
3) *Safety*: a learned-policy is coupled with an RHC controller, which in this context takes the role of a *safety filter*.
4) *Hierarchical coupling*:

Both Receding Horizon Control and Reinforcement Learning foundation of can be traced back to the work of Andrey Markov, a Russian mathematician, who developed the theory of Markov chains in the early 20th century MDPs. , , along with his colleague Howard Dreyfus, introduced MDPs as a mathematical framework for sequential decision-making under uncertainty [7]. MDPs consist of 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 $s' \in S$ according to a probability distribution $P(s' \mid s,a)$. These transition probabilities capture the stochastic nature of the environment and its dynamics. Associated with each $\{s,a\}$ pair is an immediate reward $r(s,a)$, expression of the immediate benefit or cost incurred by the decision-maker upon taking that action in that state. The objective in MDPs is to find an optimal policy $\pi(a|s)$ that maximizes the cumulative reward obtained by the decision-maker over time. This involves balancing the trade-off between immediate rewards and long-term benefits, taking into account the probabilistic nature of the environment and the dynamics of the system. In the late 1950s, Richard Bellman introduced the so-called **Bellman equation**:

Let π^* be an optimal policy for a Markov Decision Process (MDP) with state space S, action space A, state transition probabilities $P(s' \mid s,a)$, and immediate rewards $r(s,a)$. Then, for any state s in the state space S, the following equation holds:

$$V^*(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^*(s') \right]$$

where:

- *$V^*(s)$ is the optimal value function for state s,*
- *$r(s,a)$ is the immediate reward received upon taking action a in state s,*
- *$P(s' \mid s,a)$ is the probability of transitioning to state s' after taking action a in state s,*

†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.

- γ *is the discount factor representing the importance of future rewards, and*
- $\max_{a \in A}[\cdot]$ *denotes the maximum over all possible actions in state s.*

Building upon the formulation of MDPs and the above Bellman Equation, Richard Bellman and other researchers developed the theory of Dynamic Programming (DP) as a systematic method for solving optimization problems by breaking them down into simpler subproblems.

[9]
[10] [11] [12] [13] [14] [15] [16] [17] [18]
[19]
[20]
[21] [22] [23]

II. RL VERSUS RHC: *shortcomings and advantages*

III. A HYBRID APPROACH: *Learning-Based Receding Horizon Control with RL*

[**?**] [**?**]

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

[24] [25] [26] [27] [28]

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

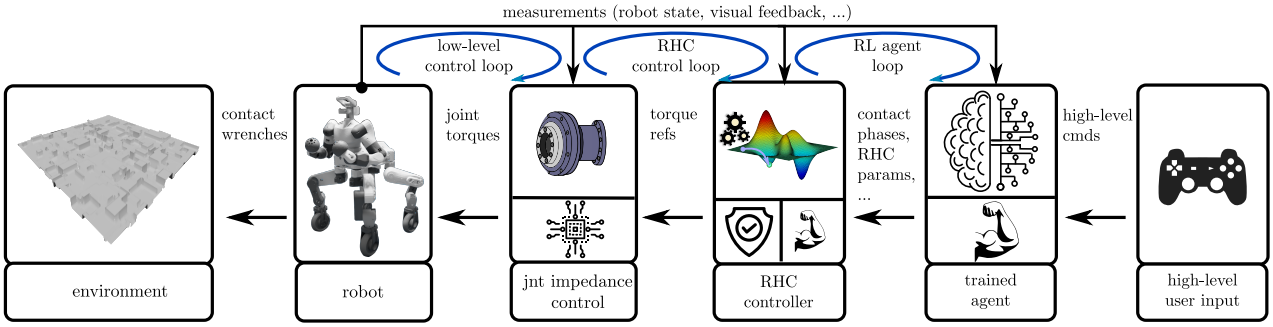

Fig. 1. Our take on Learning-based Receding Horizon Control: a MPC controller is exposed to a RL agent through key runtime parameters, like contact phases, its internal state (costs, constrains..) and interfaces for setting task commands. The agent learns to exploit the underlying RHC controller to perform 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.

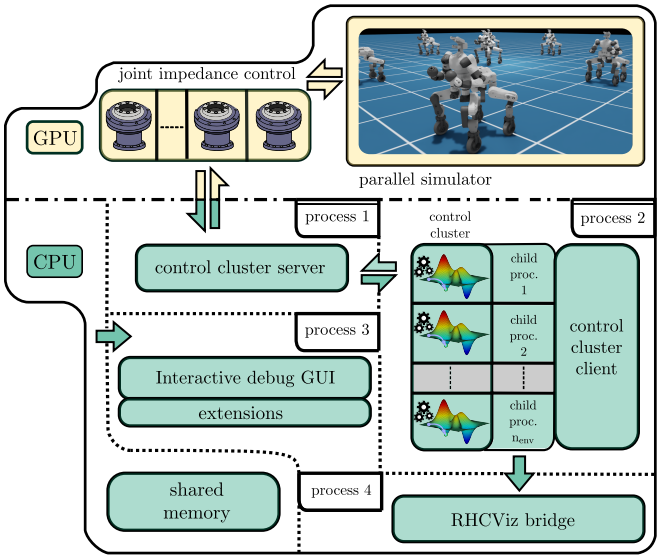

Fig. 2. High-level overview of the software implementation of the training environment to which the agent is exposed: the robot in the simulator is controlled through a joint-level impedance controller, which is in turn used by a higher-level receding horizon controller. The agent can indirectly control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

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] 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.

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

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

[9] B. F. Skinner, *The behavior of organisms: An experimental analysis*. BF Skinner Foundation, 2019.

[10] 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.

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

[12] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, pp. 280–291, Springer, 2005.

[13] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *arXiv preprint arXiv:1205.4839*, 2012.

[14] 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.

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

[16] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *International Conference on Machine Learning*, pp. 4045–4054, PMLR, 2018.

[17] 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.

[18] 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.

[19] "Mujoco 3."

[20] 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.

[21] 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.

[22] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo," dec 2022.

[23] 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.

[24] A. Patrizi, "LRHControl." https://github.com/AndrePatri/LRHControl, 2023. [Online; accessed 10-March.-2024].

[25] A. Patrizi, "OmniRoboGym." https://github.com/AndrePatri/OmniRoboGym, 2023. [Online; accessed 10-March.-2024].

[26] A. Patrizi, "CoClusterBridge." https://github.com/AndrePatri/CoClusterBridge, 2023. [Online; accessed 10-March.-2024].

[27] A. Patrizi, "RHCViz." https://github.com/AndrePatri/RHCViz, 2023. [Online; accessed 10-March.-2024].

[28] A. Patrizi, "SharsorIPCpp." https://github.com/AndrePatri/SharsorIPCpp, 2023. [Online; accessed 10-March.-2024].