# A Software Ecosystem for Research in Reinforcement Leaning-based Receding Horizon Control

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

***Abstract*— Robotics research in locomotion is undergoing a transformative shift towards the use of learning-based tools. Learning methodologies have been shown to be capable of remarkable robustness and performance even when applied to real-world environments; however, they present limitations in interpretability, safety guarantees and sample efficiency. For this reason, it is the authors' belief that more classical control approaches should not be disregarded yet. We thus advocate for a hybrid approach, combining offline data-based policy design through Reinforcement Learning (RL), with classical online Motion Planning, via 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 research 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 and approach, 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 Brief Hystorical Overview: *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 and Claude Shannon. Specifically, the introduction of the so-called *Bellman equation* [7] for the *Value Function* 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]. Later on, the development of *Policy Iteration* [8], *TD-learning* [9], *Q-learning* [10], the increased popularization of back-propagation [11] as a way of training powerful neural function approximators and the ever-increasing computational resources progressively paved the way to the ancestors of today's most successful and employed on-policy and off-policy Reinforcement Learning (RL) algorithms PPO [12] and SAC [13], respectively. Some of these ancestors include, for instance, the *Natural Policy Gradient* [14], *Deep Q-Networks* (DQN) [15], *Deep Deterministic Policy Gradient* (DDPG) [16] and *Trust Region Policy Optimization* [17] algorithms. In an analogous way, DP principles served as a foundational basis for the evolution of modern RHC [18]. Just as DP breaks down complex problems into smaller subproblems 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 receding-horizon nonlinear optimization problems have been developed, and several of them are directly tied to the continuous-time dynamics declination of DP, namely *Differential Dynamic Programming* (DDP) [19]–[22].

† 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.

## II. A hybrid approach: *Learning-Based Receding Horizon Control with Reinforcement Learning*

Most of the currently employed control tools and pipelines for locomotion rely either on online "model-based" controllers [3], [18] or "model-free" learned policies (often RL-based and trained offline) [1], [2], [23]–[35], with few exceptions [36]. In the past years there have been several attempts at combining learning based methods and receding horizon controllers, e.g. [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 [23]–[25].
2) *Adaptive tuning* and *parameter optimization*: RHC parameters tuning (e.g. weights, costs, constraints), based on real-time data [26]–[30].
3) *Safety*: a learned-policy is coupled with a RHC controller, which in this context takes the role of a *safety filter* [31]–[35].

Our approach to RL-based RHC, which is synthetically depicted in Fig. 1, can be framed as a hybrid between 1) and 2) and it is to some extent complementary to what was done in [39], where a RHC is used to rollout reference motions and footstep plans during the training of a RL tracking policy. Instead of using the RHC controller just for training, we actually aim at hierarchically coupling it with a higher level agent during both training and real-world deployment. This allows to tackle problems which are non-trivial at the RHC level (like phase selection), while also exploiting the robustness and flexibility of the agent, while maintaining the safety guarantees of the RHC controller and

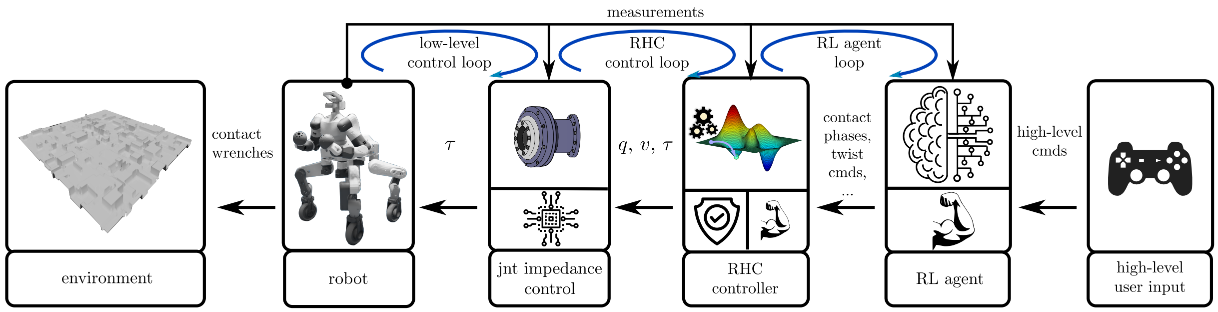


Fig. 1. Our take on Learning-based Receding Horizon Control: a RHC controller is hierarchically coupled with a higher-level RL agent during both training and real-world deployment. The RL agent has control over key RHC run-time parameters like contact phases and twist commands and can monitor its internal state (costs, constraint violations). The agent learns to exploit the underlying RHC controller to perform the tracking of user-specified high-level task references.

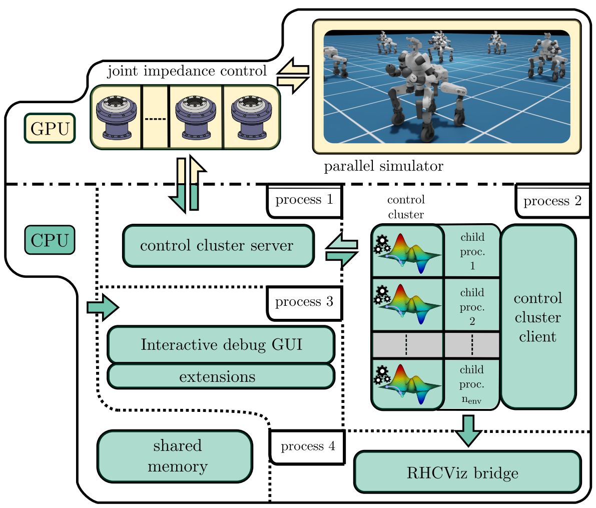


Fig. 2. High-level overview of the software implementation of the training environment to which the agent is exposed: the robot in the simulator is controlled through a joint-level impedance controller, which is in turn used by a higher-level receding horizon controller. The agent can indirectly control the robot through the latter. The training environment lives in an independent process and uses shared memory for interacting with the simulation environment.

achieving good sample efficiency. This approach, however, entails several challenges, particularly from a practical point of view (integration, computational complexity, generalization, reward formulation), which indeed make the required implementation effort non-negligible.

## III. Implementation: *framework overview and main components*

The advent of accurate GPU-accelerated simulation tools [40], [41] allows for a massive decrease in the training wall-time for data-hungry algorithms like RL and facilitates real-world deployment through domain randomization [42], [43]. We consequently choose *Omniverse IsaacSim* [40] as the simulation backend, while we use PyTorch for all deep-learning related components and *Horizon* [44] for formulating and running RHC controllers (on CPU, with an iLQR solver backend). One big drawback of this approach is the presence of the controllers on CPU, which currently represents our bottleneck in terms of training performance and parallelization capacity. Fig. 2 shows a high level software overview of the main moduli the echosystem is made of. Specifically, we developed the following software modules:

- *SharsorIPCpp* [45] serves as the shared memory backend for fast data sharing and synchronization between all components on CPU.
- *OmniRoboGym* [46] is used as a wrapper around IsaacSim and provides an interface to the *simulation* environment.
- *CoClusterBridge* [47] exploits [45] and coordinates the connection and synchronization between the simulation environment and a cluster of RHC controllers. It furthermore provides abstractions for the controllers and an extensible debug GUI for monitoring the cluster.
- *LRHControl* [48] is the main package of the echosystem and is responsible for setting up and running the simulation environment, the control cluster and the training environment.
- *RHCViz* [49] 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.

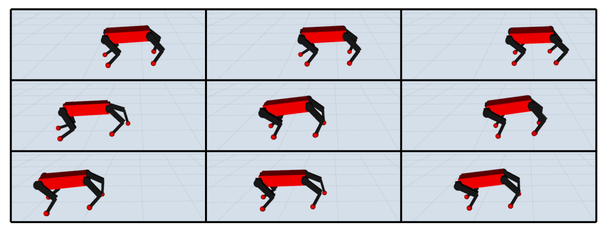

Fig. 3. Preliminary results showing the agent during learning while it moves the robot forward using the RHC controller, from right to left and from top to bottom. The corresponding video is available at [50]

## IV. A proof-of-concept example: *learning acyclic stepping for locomotion*

To showcase the potential of the proposed hybrid RL-RHC approach and of our framework, we trained a RL agent using PPO [12] to exploit a RHC controller for achieving a very simple forward locomotion task on a quadruped robot (shown in Fig. 3). The agent is given a forward velocity reference to track and is continuously rewarded based on the task error and the performance of the underlying RHC controller. Notably, we observe the emergence of completely acyclic contact phases, varying from crawling to bound-like patterns.

## 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.

[4] B. Dynamics, "Stepping Up, Reinforcement Learning with Spot." `https://www.youtube.com/watch?v=Kf9WDqYKYQQ`, 2023.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, Jan. 2024.

[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] NVIDIA, "NVIDIA Isaac Sim." `https://developer.nvidia.com/isaac-sim`.

[41] Google-DeepMind, "Mujoco 3." `https://github.com/google-deepmind/mujoco/discussions/1101`.

[42] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, pp. 91–100, PMLR, 2022.

[43] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2497–2503, IEEE, 2022.

[44] 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.

[45] A. Patrizi, "SharsorIPCpp." `https://github.com/AndrePatri/SharsorIPCpp`, 2023.

[46] A. Patrizi, "OmniRoboGym." `https://github.com/AndrePatri/OmniRoboGym`, 2023.

[47] A. Patrizi, "CoClusterBridge." `https://github.com/AndrePatri/CoClusterBridge`, 2023.

[48] A. Patrizi, "LRHControl." `https://github.com/AndrePatri/LRHControl`, 2023.

[49] A. Patrizi, "RHCViz." `https://github.com/AndrePatri/RHCViz`, 2023.

[50] A. Patrizi, "LRHControl;proof-of-concept." `proof-of-concept`.