

Planning the path with Reinforcement Learning: Optimal Robot Motion Planning in RoboCup Small Size League Environments

Mateus G. Machado, João G. Melo, Cleber Zanchettin, Pedro H. M. Braga,
Pedro V. Cunha, Edna N. S. Barros and Hansenclever F. Bassani

Centro de Informática, Universidade Federal de Pernambuco.
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - Pernambuco, Brazil.
{mgm4, jgocm, cz, phmb4, pvc3, ensb, hfb}@cin.ufpe.br

Abstract. This work investigates the potential of Reinforcement Learning (RL) to tackle robot motion planning challenges in the dynamic RoboCup Small Size League (SSL). Using a heuristic control approach, we evaluate RL’s effectiveness in obstacle-free and single-obstacle path-planning environments. Ablation studies reveal significant performance improvements. Our method achieved a **60% time gain** in obstacle-free environments compared to baseline algorithms. Additionally, our findings demonstrated **dynamic obstacle avoidance capabilities**, adeptly navigating around moving blocks. These findings highlight the potential of RL to enhance robot motion planning in the challenging and unpredictable SSL environment.

Keywords: Reinforcement Learning, Motion Planning, Robot Control

1 Introduction

The RoboCup Small Size League (SSL) is a robot soccer competition that focuses on the problem of multi-robot cooperation and control in a highly dynamic environment [4]. This challenging game requires controlling a team of up to 11 robots to work together to perform passes, dribbles, and shots on goal in a coordinated and effective way. The effectiveness of these actions hinges on precise path-planning tailored to the distinct motion models of each robot [10].

Figure 1 represents the general workflow for planning a robot’s motion within the SSL environment. Given a target position and orientation in the environment (depicted in orange), the process unfolds in three key steps. Initially, global planning calculates a geometric path, represented by a set of coordinates in the configuration space, guiding the robot toward the desired target. Subsequently, the local planner generates a sequence of desired states, comprising positions and velocities, forming a trajectory aligning with the previously computed path. Finally, the motion control module determines the requisite control inputs for the robot to transition from its current state to the next desired state.

Reinforcement Learning (RL) has emerged as a promising approach for robotic control tasks[16,19]. RL offers a paradigm where agents learn optimal strategies

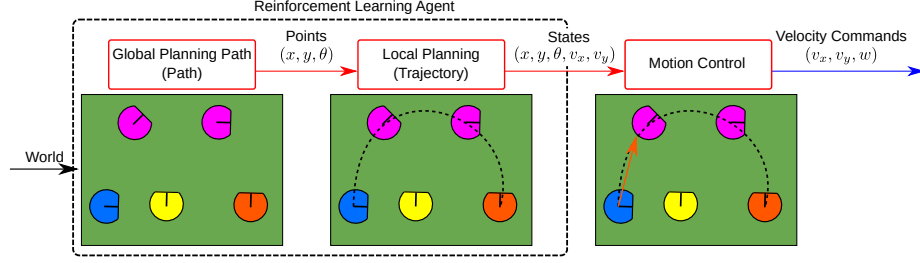


Fig. 1: General workflow for robot motion planning and control, divided into Global planning, local planning, and motion control. The first two stages plan the points and velocities the agent should follow, while the last controls the robot. This Work focuses on the first two modules of this workflow.

through interaction with their environment. This learning paradigm finds a natural application in the SSL, as it allows agents to adapt and improve their decision-making processes in response to the dynamic and unpredictable nature of the game [29].

We discern a hierarchical structure by adapting the motion planning workflow to RL learning environments. While the motion control module directly governs the robot’s actions, the global and local planning modules serve as intermediary sub-goals to be achieved. We refer to the motion control task as **goToPoint**, with global and local planning denoted as **path-planning**.

Building upon this foundation, we propose a model-free path-planning methodology, leveraging goal-conditioned policies that take the goal state as input [14,26], and yield sub-goals to be executed by an omnidirectional motion control [28]. Sim2Real is a subfield of Machine Learning focused on mitigate the gap between simulated and real-world environments [13,9]. While RL environments traditionally model the entire motion planning system, integrating a Sim2Real module becomes imperative to transfer learned models to real-world settings. Our methodology circumvents this challenge, as the path-planning model remains agnostic to the training environment, whether simulated or real.

Our primary contribution lies in a methodology for employing Reinforcement Learning in the path-planning task, creating a model that seamlessly transitions from simulation to reality [10,11]. We evaluate the efficacy of the Soft Actor-Critic (SAC) [12] algorithm across both baseline and proposed learning environments, shedding light on discrepancies inherent in the baseline’s reliance on goToPoint to address the path-planning task. Additionally, we introduce two state-of-the-art methods to mitigate action instability and craft intuitive trajectories. Finally, through empirical validation, we demonstrate the readiness of our model for real-world deployment.

2 Related Work

Traditionally, motion planning systems are conceptualized as single complex agents aimed at solving the entire system [7,8,22]. However, a hierarchical understanding of the system [23,30] delineates a manager-worker structure, with the path-planning task delegated to a manager agent and `goToPoint` executed by a worker agent. In our approach, we base the environment development on the managerial treatment of the environment in existing works.

In Reinforcement Learning, the reward function quantifies the impact of an action on the environment and serves as a crucial component to guide the agent through the learning process [29]. Our baseline work [5] endeavors to solve the path-planning task for robots in the Small Size League (SSL) of RoboCup [4]. As elucidated in subsequent sections, [5] adopts a reward system emphasizing direct impact actions, culminating in an inherently unstable `goToPoint` agent.

The challenge of ensuring the robustness of RL algorithms in handling continuous actions has been extensively studied in the realm of robot learning [18,25,27]. A hallmark of a proficient path-planning algorithm lies in its ability to generate a minimal set of stable output points. Contrarily, algorithms such as the one proposed in [5] exhibit instability, rendering them impractical for real-world deployment. To address this, we employ two learning methods for the agent [15,21], to produce a model characterized by desired features such as action stability and simplicity.

Our methodology establishes a conducive learning environment for the path-planning task, anchored on three core pillars: a reward system centered on agent actions, action stability, and simplicity in action selection. Our results yield path-planning agents capable of generating intuitive trajectories for human comprehension. While the motion control aspect for SSL robots has been extensively explored in prior works [3,2,1], our deliberate abstraction of this module enables our agents to seamlessly integrate into real-world setups, facilitating a plug-and-play deployment paradigm.

3 Motion Planning Learning Environments

We adopted a similar environment structure guided by the fundamental principles of [5]. However, we explore RL-based path-planning to abstract the motion control module, following the pipeline presented in Figure 1 ¹.

3.1 Baseline Environment

Cruz *et al.* [5] explore two RL-based motion planning approaches: RL-based path-planning to a given control system and a single-component approach, where the RL agent learns both motion planning and control. As a starting point for

¹ Agents, both trainable and trained, are available at <https://github.com/goncamateus/Planning-the-path-with-rl>

this work, we have successfully replicated their environments within the rSoccer framework [19], employing the first approach as a Baseline environment for this research. The state space within these environments is characterized by a 13-dimensional vector: $(x^t, y^t, \cos(\theta^t), \sin(\theta^t), v_x^t, v_y^t, x^r, y^r, \cos(\theta^r), \sin(\theta^r), v_x^r, v_y^r, \omega^r)$, where t and r labels account for target and robot respectively. Also, in applying RL-based path-planning to a given motion control, the action space consists of a 6-tuple $(x, y, v_x, v_y, \sin \theta, \cos \theta)$, generating a sub-goal at each environment step. Notably, these vectors are subject to normalization concerning the maximum values in each dimension, except for the angular components, represented as sine and cosine values.

The reward system in [5] is defined by $R_T(s, a) = R_d(s, a) + R_\theta(s, a) + R_t(s, a)$, where:

$$\begin{aligned} R_d(s, a) &= \begin{cases} -d(s, a), & \text{if } \text{dist}(s, a) > D_{th} \\ 10, & \text{otherwise} \end{cases} \\ R_\theta(s, a) &= \begin{cases} \frac{-\delta(s, a)}{\pi}, & \text{if } \delta(s, a) > \theta_{th} \\ 1, & \text{otherwise} \end{cases} \\ R_t(s, a) &= \begin{cases} 1000, & \text{if } \text{dist}(s, a) > D_{th} \text{ and } \delta(s, a) > \theta_{th} \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Here, s and a are state and action, $d(s, a)$ and $\delta(s, a)$ are the distance and angle difference to the target, with thresholds D_{th} and θ_{th} , respectively. As elucidated in Section 2, these rewards do not depend directly on the actions taken but on the robot's actuators. See in Figure 2a a graphical representation of the angle difference and the distance to the target.

3.2 Proposed Environment

Firstly, we simplified the learning problem by imposing two main constraints: the agent's actions velocity components magnitude is constrained to zero; the goal also has the target velocity magnitude constrained to zero. Therefore, the Local Planning, as seen in Figure 1, is in form of $x, y, \theta, 0, 0$. The state and action spaces remain the same as in the Baseline.

The reward system proposed by [5] accounts only for the robot's properties, underscoring the agent's responsibility to optimize its actions and achieve the desired objectives. We introduced an action-centric reward structure by adding action features to our reward system, emphasizing states where the robot is on the cusp of achieving favorable outcomes within each reward component.

Therefore, the whole reward system, R_d, R_θ , and R_t , outlined in Section 3.1 remains the same but changes the reference from the robot to the sub-goal action of our path-planning agent. Note in Figure 2b a visual difference between our Proposed environment and the Baseline.

In our proposed reward system, the agent is penalized if the generated path is incompatible with the target. This strategic shift ensures that the agent's learned

actions align more closely with the desired trajectory, fostering a genuine path-planning capability and reinforcing the agent’s focus on optimal and effective target-reaching strategies.

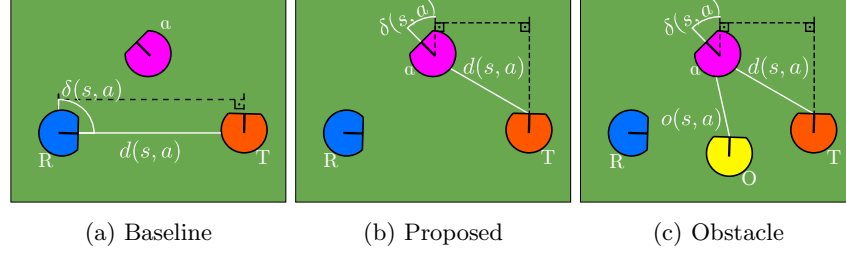


Fig. 2: Visualization of the “Baseline”, “Proposed”, and “Obstacle” learning environments with example rewards. The circles blue (R), magenta (a), yellow (O), and orange (T) represent the robot’s position, the action taken, the obstacle, and the target goal, respectively. The white arc and lines illustrate the angular difference, $\delta(s, a)$, and distances, $d(s, a)$, $o(s, a)$, to target and obstacle, respectively.

3.3 Proposed Obstacle Environment

The obstacle setup inherits the Proposed environment and adds an enemy robot as the obstacle. Therefore, the state space for this environment is incremented by a 5-dimensional vector referent from the obstacle: $x^o, y^o, v_x^o, v_y^o, \omega^o$. The obstacle moves randomly, varying its speed in a range of $[0, 1]m/s$ each step. In this environment, we also add penalties for hitting the obstacle and the actions being close to it. When the agent hits an obstacle, the episode fails, and the penalty is the inverse of reaching the target.

At each step, we penalize the agent’s proximity from the obstacle with a parametric Gaussian in function of the distance between the action and the obstacle $o(s, a)$, expected value $\mu = 0$, and variation $\sigma = 1$. Therefore, we can describe the reward system as $R_T(s, a) = R_d(s, a) + R_\theta(s, a) + R_t(s, a) + R_{obst}(s, a) + R_{hit}(s, a)$, where $R_{hit}(s, a) = -1000$ if the robot hits the obstacle; otherwise, 0. See the addition of the obstacle distance reward in Figure 2c.

4 Mitigating Action Instability

The prominent issue of erratic and non-smooth actions learned by RL policies is well-recognized, posing potential concerns such as overheating and energy wastage problems in practical applications [17, 6]. In our context, this problem manifests as the agent controlling the robot not through angular or linear velocities but via poses. We introduce two techniques known to be effective in addressing this challenge in our training process.

The *Frame skip* wrapper simplifies action sequences by repeating a given action for a specified number of steps. As it demonstrated to enhance agent performance in Atari games using Deep Q-Networks [20,15], we apply this wrapper with a skip rate of 16 in the rSoccer framework [19]. This adjustment significantly reduces the number of actions per episode from 1200 to a maximum of 75.

Mysore *et al.* [21] introduce an intuitive regularization technique, achieving an impressive 80% reduction in power consumption during robot experiments. This technique, termed *Conditioning for Action Policy Smoothness (CAPS)*, introduces an additional loss term for the Actor in an Actor-Critic setup. CAPS aims to penalize actions that exhibit significant differences for similar states, promoting smoother and more controlled behavior.

5 Results

We evaluate the agent’s performance in the presented environments, namely Baseline, Proposed, and Obstacle, comparing their episodic lengths and the Cumulative Pairwise Action Distance (**CPAD**), a novel proposed measure designed to assess the stability of executed actions. This metric quantifies the overall displacement solely in the (x, y) position and is mathematically defined as: $D(A) = \sum_{i=0}^{K-1} \text{Dist}(A_i, A_{i+1})$, where A represents the actions undertaken throughout the episode, and K denotes the episode length.

CPAD offers valuable insights into the agent’s behavioral stability. Specifically, a consistently directed focus on a particular (x, y) position results in a cumulative distance of 0. Conversely, rapid and erratic movements yield higher cumulative distances. As such, a lower CPAD is a crucial indicator of more stable and consistent actions, a pivotal attribute in motion planning.

Next topics unveils the results obtained from trained agents using the Soft Actor-Critic (**SAC**) [12] method in the setups described in Section 3. It is important to note that, given our use of the SAC algorithm for training agents, adjustments were made to accommodate the CAPS algorithm. SAC’s policy entropy factor α is a parameter of the policy learning method to induce exploration during training. This parameter is originally set as trainable to, for example, explore at the beginning of training and then having stable actions. When applying CAPS at SAC, the α parameter hinds the stability loss, and the learning in general. Therefore, we set α as a constant throughout the SAC-CAPS algorithm execution.

5.1 Comparing Baseline and Proposed Environments

This evaluation assesses SAC agents in both Baseline and Proposed environments using four different setups: **1)** Vanilla, **2)** with Frame Skip, **3)** with CAPS, and **4)** with both Frame Skip and CAPS (FSCAPS). These setups provide insights into the model’s adaptability within a real-world environment.

Table 1 shows the proposed metrics results for obstacle-free environments. Notably, both metrics gradually decrease with the addition of action-stabilizing methods. The Proposed environment consistently outperforms the Baseline, showcasing improved agent motion.

Table 1: Test results distributions are presented for obstacle-free environments over 1000 episodes. Pairwise comparisons are conducted for each setup using the methods outlined in Section 4. The Vanilla experiments involve running agents in the environments described in Section 3 without any action stabilizing method. The Frame Skip and CAPS experiments are similar but include either the frame skip or CAPS method. The FSCAPS experiments encompass both stabilization methods.

	Environment	Episode Length (steps)			CPAD (m)		
		IQR	Max	Min	IQR	Max	Min
Vanilla	Baseline	1200 (1200 - 1200)	1200	52	20.84 (14.42 - 26.75)	47.66	1.39
	Proposed	128 (102 - 152)	1200	35	6.03 (3.93 - 8.07)	16.43	0.20
Frame Skip	Baseline	198 (150 - 265)	1200	49	13.53 (11.03 - 16.55)	38.40	3.46
	Proposed	119 (95 - 141)	223	33	1.31 (0.60 - 2.12)	6.80	0.01
CAPS	Baseline	1200 (731 - 1200)	1200	37	13.97 (9.61 - 16.31)	22.86	1.91
	Proposed	113 (91 - 134)	1200	33	1.29 (0.81 - 1.87)	3.81	0.14
FSCAPS	Baseline	109 (87 - 130)	196	34	0.58 (0.36 - 0.87)	2.22	0.02
	Proposed	114 (92 - 137)	193	31	0.06 (0.05 - 0.08)	0.16	0.00

When evaluating individual experiments through a “Baseline *vs* Proposed” lens, our proposed environment consistently demonstrates superior performance in both episode length and CPAD metrics. The Vanilla experiment, notable improvements of **90% for episode length** and **71% for CPAD** were observed. Incorporating the Frame Skip wrapper yielded a significant performance enhancement, reducing task completion time by approximately **80 steps** (equivalent to 6% of the total episode) and achieving a **90%** improvement in the CPAD metric. In the CAPS experiment, inspired by [21], we achieved a **78%** CPAD reduction compared to Vanilla-Proposed. This outcome underscores the limitations of baseline-trained agents in attaining the precision of our proposed approach.

Combining action-stabilizing methods in the FSCAPS experiments showcases enhanced performance. The Baseline-trained agent exhibits marginally improved episode length, while the Proposed-trained agent demonstrates a remarkable **tenfold improvement** in action precision. Therefore, the gain in episode length is deemed negligible compared to the Proposed agent’s superior transfer capability to a real-world environment.

In comparing experiments within our proposed environment, the final temporal gain between Vanilla and FSCAPS experiments is modest, with only 14 simulated steps (1% of the total episode) faster. However, examining the CPAD metric reveals a substantial increase in action stability with each addition, resulting in a final gain **100 times better** than the environment alone. See in

Figure 3 trained agents for our proposed environment in each experiment. These compelling results encourage further research, particularly in environments with obstacle setups.

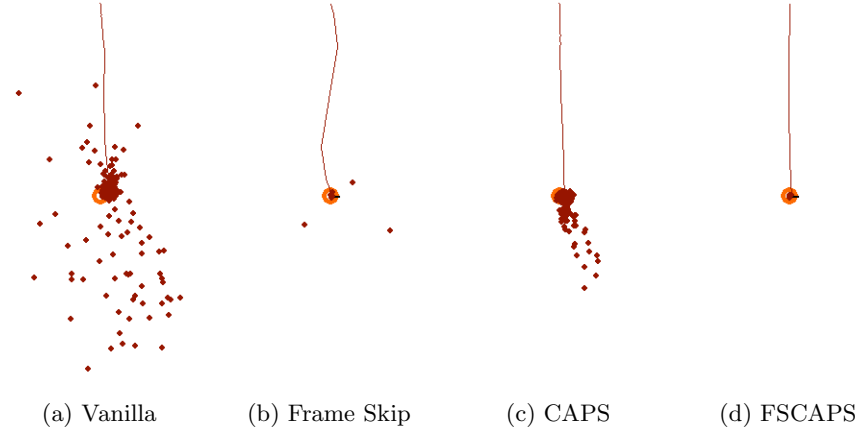


Fig. 3: Trained Soft Actor-Critic agents operating within the Proposed environment across various experiments. The orange circle denotes the target goal, while the red dots trace the actions executed by the agent throughout the episode. The red line visually maps the trajectory followed by the robot. As depicted in Figure 3d, the agent demonstrates precision by consistently reaching the target without deviation.

5.2 Evaluating Obstacle Avoidance

The proposed Obstacle environment is evaluated using SAC agents following the scheme outlined in Section 5.1. This environment features a simple scenario with a single mobile robot acting as the obstacle. Uniquely in this setup, the percentage of hitting the obstacle is introduced as an additional metric, providing insights into the adaptability of our previous research in obstacle-laden environments.

Table 2 presents results from experiments in the Obstacle environment. Preliminary results indicate that the baseline method struggled to solve the task consistently within the imposed training time. In contrast, our final method, FSCAPS, introduces a robust and adaptable approach for environments with obstacle. Understandably, variations in actions are observed as the agent needs to navigate around the moving obstacle. The low rate of collisions suggests that the agent effectively learned the purpose of the environment, even at maximum speed, with the episode length nearly identical to the setup without obstacles.

Table 2: Test results distributions are presented for the environment with obstacle over 1000 episodes.

	Episode Length (steps)			CPAD (m)			Collision
	IQR	Max	Min	IQR	Max	Min	
Vanilla	1200 (1200 - 1200)	1200	132	12.38 (10.70 - 14.21)	35.03	4.10	12.38%
Frame Skip	1200 (1200 - 1200)	1200	148	6.03 (5.24 - 6.90)	11.35	2.23	5.61%
CAPS	1200 (1200 - 1200)	1200	670	11.35 (10.55 - 12.17)	20.60	4.10	1.32%
FSCAPS	115 (113 - 158)	1200	79	1.27 (1.06 - 1.53)	5.87	0.55	0.61%

5.3 Real environment results

After obtaining stable and accurate results in a simulated environment, the trained agents were subsequently deployed in a real-world setting. We integrated our model into the RobôCIn SSL-Unification software [3] using the TorchScript module [24] for compilation and adaptation purposes. It is important to note that the RobôCIn SSL-Unification software employs its own motion control system, whereas our agents were trained using a general omnidirectional motion control approach [28].

The agents utilized in these experiments included the Vanilla agent, trained in an obstacle-free environment, and the FSCAPS agent, trained in our proposed “Obstacle” environment. For experiments involving obstacles, we employed a fixed obstacle to ensure safety.

Table 3 presents the episode length and CPAD for the experiments on the real-world. The Vanilla agent failed to complete the task in any of the ten trials. The CPAD metric suggests unstable paths extending to a length of 25 fields, turning it unpractical to real-world.

In contrast, the FSCAPS agent exhibited consistent results across both environment setups. These findings align with our approach of developing a methodology that is agnostic to the motion control system, thereby facilitating deployment in real-world scenarios.

Table 3: Test results distributions are presented for real-world setup over 10 episodes.

	Episode Length (steps)			CPAD (m)		
	IQR	Max	Min	IQR	Max	Min
Vanilla	1200 (1200 - 1200)	1200	1200	227.93 (227.06 - 230.79)	235.06	225.45
FSCAPS	159 (155 - 160)	162	135	0.10 (0.10 - 0.10)	0.99	0.09
Obstacle	140 (131 - 151)	171	109	2.10 (2.04 - 2.15)	2.32	2.01

6 Discussion

In analyzing agent adaptability, FSCAPS demonstrates significant superiority in acquiring efficient strategies in challenging environments. The combination of Frame Skip with CAPS accelerates adaptation to obstacle dynamics, evidenced by reduced episode durations and collision rates. This suggests more effective learning and better generalization compared to isolated methods. Environmental parameters significantly impact path planning efficacy, with FSCAPS exhibiting greater robustness against environmental variations. This emphasizes the need for strategies resilient to fluctuations.

Compared to state-of-the-art approaches, FSCAPS not only outperforms in efficiency and safety but also strikes a balanced solution, especially in complex obstacle environments where real-time adaptation is crucial. Navigation failures revealed that Pure and Frame Skip methods struggled with unpredictable obstacles. The integration of CAPS enhances predictability and action smoothness, reducing collision rates.

While longer episodes offer more experience, they can induce fatigue. FSCAPS reduces episode duration without compromising safety, offering a sustainable training approach. However, generalizing FSCAPS to untested scenarios requires further exploration to validate its real-world robotic applications.

7 Conclusion

In this study, we comprehensively evaluated Reinforcement Learning (RL) applied to the intricate challenges of robot motion planning within the RoboCup Small Size League (SSL). Leveraging a heuristic control approach, we explored various learning environments, comparing the performance of SAC agents in both Baseline and Proposed setups.

Our exploration in obstacle-free environments showcased the Proposed environment’s effectiveness in reducing action instability, and enhancing agent performance. The Frame Skip wrapper further improved efficiency, and the combined use of Frame Skip and CAPS in the “FSCAPS” experiments demonstrated significant performance gains.

Transitioning to obstacle-laden environments, the initial experiments faced challenges in consistently solving tasks, emphasizing the complexity introduced by mobile obstacles. However, our final method, “FSCAPS”, exhibited evident adaptability, effectively navigating around obstacles with a minimal rate of collision.

Furthermore, we validated the adaptability of our model in a real-world setting without compromising performance in terms of trajectory length or precision. This corroborates the efficacy of our approach, emphasizing the importance of initially addressing high-level tasks for seamless model adaptation between simulation and real-world deployment.

These results underscore the significance of our research in advancing RL techniques for robot motion planning, particularly in dynamic and challenging

SSL environments. The strategic integration of heuristic control and innovative stabilizing techniques demonstrates the potential for improved decision-making and adaptability in dynamic robotic scenarios, paving the way for advancements in the field of autonomous robotics.

References

1. Abiyev, R.H., Akkaya, N., Gunsell, I.: Control of omnidirectional robot using z-number-based fuzzy system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **49**(1), 238–252 (2018)
2. Abiyev, R.H., Gunsell, I.S., Akkaya, N., Aytac, E., Çağman, A., Abizada, S.: Fuzzy control of omnidirectional robot. *Procedia Computer Science* **120**, 608–616 (2017)
3. Araújo, V., Rodrigues, R., Cruz, J., Cavalcanti, L., Andrade, M., Santos, M., Melo, J., Oliveira, P., Morais, R., Barros, E.: Robôcin ssl-unification: A modular software architecture for dynamic multi-robot systems. In: *Robot World Cup*, pp. 313–324. Springer (2023)
4. Committee, S.S.L.T.: Rules of the RoboCup Small Size League 2023, <https://robocup-ssl.github.io/ssl-rules/sslrules.pdf>
5. Cruz, J.V.S.: Exploring reinforcement learning in path planning for omnidirectional robotsoccer (2023), <https://repositorio.ufpe.br/handle/123456789/51592>
6. Duan, Y., Chen, X., Houthoofd, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. *CoRR* **abs/1604.06778** (2016), <http://arxiv.org/abs/1604.06778>
7. Eysenbach, B., Salakhutdinov, R., Levine, S.: Search on the replay buffer: Bridging planning and reinforcement learning (2019)
8. Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J., Zhang, T.: Dher: Hindsight experience replay for dynamic goals. In: *International Conference on Learning Representations* (2018)
9. Gao, R., Si, Z., Chang, Y.Y., Clarke, S., Bohg, J., Fei-Fei, L., Yuan, W., Wu, J.: Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 10598–10608 (June 2022)
10. Gasparetto, A., Boscariol, P., Lanzutti, A., Vidoni, R.: *Path Planning and Trajectory Planning Algorithms: A General Overview*, pp. 3–27. Springer International Publishing, Cham (2015)
11. González, D., Pérez, J., Milanés, V., Nashashibi, F.: A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* **17**(4), 1135–1145 (2016). <https://doi.org/10.1109/TITS.2015.2498841>
12. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft actor-critic algorithms and applications. *CoRR* **abs/1812.05905** (2018), <http://arxiv.org/abs/1812.05905>
13. Hu, X., Li, S., Huang, T., Tang, B., Huai, R., Chen, L.: How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence. *IEEE Transactions on Intelligent Vehicles* **9**(1), 593–612 (2024). <https://doi.org/10.1109/TIV.2023.3312777>
14. Kaelbling, L.P.: Learning to achieve goals. In: *IJCAI*. vol. 2, pp. 1094–8. Citeseer (1993)

15. Kalyanakrishnan, S., Aravindan, S., Bagdawat, V., Bhatt, V., Goka, H., Gupta, A., Krishna, K., Piratla, V.: An analysis of frame-skipping in reinforcement learning. CoRR **abs/2102.03718** (2021), <https://arxiv.org/abs/2102.03718>
16. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), <http://arxiv.org/abs/1509.02971>
17. Mahmood, A.R., Korenkevych, D., Vasan, G., Ma, W., Bergstra, J.: Benchmarking reinforcement learning algorithms on real-world robots. CoRR **abs/1809.07731** (2018), <http://arxiv.org/abs/1809.07731>
18. Mankowitz, D.J., Levine, N., Jeong, R., Shi, Y., Kay, J., Abdolmaleki, A., Springenberg, J.T., Mann, T., Hester, T., Riedmiller, M.: Robust reinforcement learning for continuous control with model misspecification. arXiv preprint arXiv:1906.07516 (2019)
19. Martins, F.B., Machado, M.G., Bassani, H.F., Braga, P.H.M., Barros, E.S.: rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: Alami, R., Biswas, J., Cakmak, M., Obst, O. (eds.) RoboCup 2021: Robot World Cup XXIV. pp. 165–176. Springer International Publishing, Cham (2022)
20. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR **abs/1312.5602** (2013), <http://arxiv.org/abs/1312.5602>
21. Mysore, S., Mabsout, B., Mancuso, R., Saenko, K.: Regularizing action policies for smooth control with reinforcement learning. CoRR **abs/2012.06644** (2020), <https://arxiv.org/abs/2012.06644>
22. Nair, A.V., Pong, V., Dalal, M., Bahl, S., Lin, S., Levine, S.: Visual reinforcement learning with imagined goals. Advances in neural information processing systems **31** (2018)
23. Narasimhan, K., Osband, I., Scherr, R., Xing, Y., Swearingen, J., Li, C., Tan, J., Wainwright, M.: Feudal reinforcement learning. arXiv preprint arXiv:1811.03244 (2018)
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019)
25. Perrusquía, A., Yu, W.: Continuous-time reinforcement learning for robust control under worst-case uncertainty. International Journal of Systems Science **52**(4), 770–784 (2021)
26. Pong, V., Gu, S., Dalal, M., Levine, S.: Temporal difference models: Model-free deep rl for model-based control. arXiv preprint arXiv:1802.09081 (2018)
27. Rahul, M., Chiddarwar, S.S.: Deep reinforcement learning with inverse jacobian based model-free path planning for deburring in complex industrial environment. Journal of Intelligent & Robotic Systems **110**(1), 4 (2024)
28. Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D.: Introduction to autonomous mobile robots. MIT press (2011)
29. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. The MIT press (2018)
30. Sutton, R.S., Precup, D.: Learning multi-level hierarchies with hindsight. In: International Conference on Machine Learning. pp. 861–868 (1998)