Open-Source Foosball Benchmark for Deep Reinforcement Learning

Matthew So [*1] **Kwansoo Lee** [*2] **Judah Goldfeder** [1] **Hod Lipson** [1 2]

Abstract

Foosball is a fast-paced, strategy-driven table game that requires sub-second decision-making, fine motor control, and dynamic tactics. Reinforcement learning (RL) algorithms are widely used for AI agents to implicitly learn the physical world. In this work, we present a new open-source framework designed for evaluating end-to-end deep RL algorithms in a simulated foosball environment. Our platform includes a high-fidelity physics simulation environment built in MuJoCo, enabling analysis of performance transfer from virtual to real-world conditions. This platform is designed to advance the development of competitive agents capable of robust, adaptive behavior and effective sim-to-real generalization, serving as a standardized resource for the broader RL community. Code available at: https://github.com/thakur-sachin/Foosball_CU

1. Introduction

Foosball is a fast-paced, strategy-driven table game that requires sub-second decision-making, fine motor control, and dynamic tactics. Reinforcement Learning (RL) techniques are widely adopted for automating dynamic control tasks because they can discover near-optimal policies across diverse environments (Margolis et al., 2022; Tang et al., 2024; Kober et al., 2013). To effectively train the fast and physical skills involved in foosball, an RL agent must learn within a simulator that faithfully reproduces the sport's real-world physics and high-speed dynamics, ensuring the resulting policy can reliably transfer to an actual table with robustness (D'Ambrosio et al., 2023; Kaufmann et al., 2023). Furthermore, choosing an RL algorithm well suited to these rapid, interactive settings is essential for maximizing decision-making performance (Kurach et al., 2020; Tammewar et al., 2023).

In this work, we introduce an end-to-end pipeline that enables researchers to train agents in simulation and evaluate them on standardized physical hardware. The system includes a high-fidelity physics simulation model, a real-world foosball table with controllable actuators and RGB cameras, and a global leaderboard that spans both simulated and real-world performance in order to explore the domain of sim-to-real transfer in foosball playing.

The pipeline is set up to feed raw images directly into a Deep RL Agent, eliminating intermediate steps such as ball localization or rod-angle estimation. We propose this as a physical foosball platform that can be used to compare RL agents trained in separate simulated environments. Our preliminary benchmarks consist of three reinforcement-learning algorithms: Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018), Proximal Policy Optimization (PPO) (Schulman et al., 2017), and Soft Actor-Critic (SAC) (Haarnoja et al., 2019). We measure how reliably each can score when domain randomization is applied and training resources are held constant. We conclude with a discussion of the benchmark's potential for advancing sim-to-real research in competitive multi-agent settings.

2. Related Work

Research on autonomous foosball spans both simulated settings and physical tables. The studies summarized below highlight key progress and open challenges.

2.1. Game-State Estimation

Several projects extract the match state from camera images of the foosball surface (Gutierrez-Franco et al., 2013; Hernández et al., 2019; Weigel & Nebel, 2003). Janssen *et al.* showed that a combination of regions of interest (ROI) and Kalman filtering can reliably track the ball in a known arena at speeds up to 10 m/s—even with rebounds and sudden direction changes (Janssen et al., 2009). More recently, CNN-based approaches have inferred rod positions and ball locations directly from images, underscoring deep

[*]Equal contribution [1]Computer Science, Columbia University, New York, USA [2]Department of Mechanical Engineering, Columbia University, New York, USA. Correspondence to: Matthew So <ms5513@columbia.edu>, Kwansoo Lee <kl3563@columbia.edu>, Judah Goldfeder <jag2396@columbia.edu>, Hod Lipson <hod.lipson@columbia.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

networks' capacity to recover a foosball game state without hand-crafted features (Hagens et al., 2024; Horst et al., 2024).

2.2. Action Selection

Early autonomous players relied on hand-tuned decision trees for rod manipulation (Weigel & Nebel, 2003; Weigel, 2005). Imitation-learning pipelines soon followed, capturing advanced moves such as the "lock" and "slide-kick" techniques (Zhang & Nebel, 2007). With the rise of Deep Reinforcement Learning (DRL), researchers framed scoring with a single rod as a control problem akin to a manufacturing cell (De Blasi et al., 2020). Subsequent studies extended DRL to individual skills such as dribbling and passing (Croenen et al.), and to multi-agent scenarios in which strikers and goalkeepers learn competing offensive and defensive policies from scratch (Gashi et al., 2023; Pinto et al., 2017).

2.3. Sim-to-Real Transfer

Many works build physical foosball robots and then deploy the policies trained in simulation on the real hardware (Moos et al., 2024; De Blasi et al., 2020; Weigel, 2005; Rohrer et al., 2021). This line of research sits within the broader Sim-to-Real literature for robotic manipulation (Peng et al., 2017a;b; Tobin et al., 2017). Reports consistently note a marked performance drop when control policies leave the simulator (Moos et al., 2024; Rohrer et al., 2021). Domain randomization—systematically varying simulation parameters—has been proposed as a hedge against overfitting to the virtual environment (De Blasi et al., 2020; Peng et al., 2017b).

3. Building The Simulation Environment

Foosball play involves rapid interactions between the ball and players. Given the features of fast movements of the ball and accurate contact state estimation during the play, MuJoCo (Multi-Joint dynamics with Contact) physics simulator is utilized to conduct the simulation training in this study.

MuJoCo is an open-source physics engine designed to facilitate advanced scientific research and development in domains such as robotics, biomechanics, graphics, and animation. The simulator provides various constraints and configurable options for contact and friction dynamics, enabling accurate contact estimation. Its capacity for both precise and rapid simulations makes it particularly well-suited for applications that require accuracy and computational efficiency, such as foosball playing (Todorov et al., 2012).

The foosball table model used is the Tornado T-3000 Foosball Table, which is the official table of the International Table Soccer Federation (ITSF). As shown in Figure 1, the

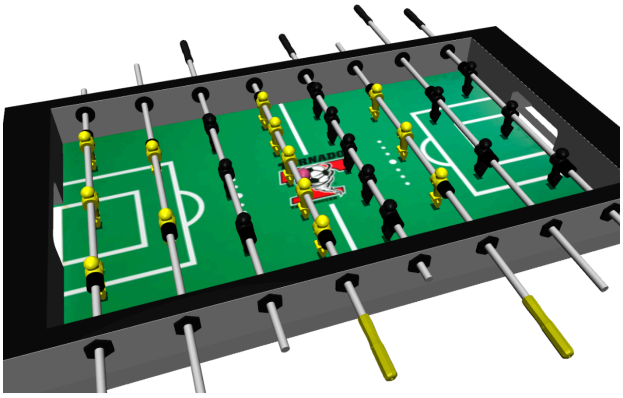


Figure 1: MuJoCo Foosball Simulation Environment

3D the foosball table's components are reconstructed in the MuJoCo simulation environment using Signed Distance Field (SDF).

There are three constraint solver parameters under user's control, which determine the behaviors of the constraint models in the simulation. These parameters can be set indirectly through the attributes available in MJCF (MuJoCo XML Format) elements. A new constraint formulation is used in MuJoCo to define the constraint. To embody accurate contacts and fast calculation of collision, we construct a strong constraint model by setting impedance the solver parameters. In addition, to ensure rapid recovery from constraint violations, we set the time constant to a small value. We are further exploring the parameters, contact models, and friction loss to achieve the realistic movement of the ball.

3.1. Episodic Setup

In our experiments, the agent is given full control over the yellow foosball rods and is incentivized to score in the goal guarded by the black opponents. The center of the intended goal is characterized by x and y coordinates (0,64) in the environment setup.

Observation Space

Our framework supports the use of an image-based observation space or a position-based space. In all of these exploratory experiments, the agent is given the 38-dimensional position-based space, comprised of the linear position of all rods (8), the rotational position of all rods (8), the linear velocity of all rods (8), the rotational velocity of all rods (8), 3-dimensional position of the ball (3) and 3-dimensional velocity of the ball (3).

ACTION SPACE

In all experiments, the agent is given linear and rotational control over all joints for one side of play. This results in an 8-dimensional action space.

REWARD FUNCTION

Several iterations of a reward function were attempted during this study. Approaches that included rod movement penalties, ball velocity, time till scoring, etc. were tried. In practice, the reward function yielding the best results was one that simply incentivized movement towards the goal and an increased reward when a goal was achieved. The following reward function was used:

$$\alpha \; - \; \sqrt{(x_b - x_g)^2 + (y_b - y_g)^2} \; + \; \beta\, \mathbb{I}\{y_b \geq y_g\}$$

where x_b and y_b are the x and y components of the ball, x_g and y_g are the x and y components of the goal, and α and β are tunable parameters. During the training, $\alpha = 300$ and $\beta = 1000$.

EPISODE

Every episode begins with all rods in their starting position and the ball in a slightly randomized position. The x and y coordinates are drawn independently as follows:

$$x_b \sim \mathcal{N}(-0.5, 0.5^2), \quad y_b \sim \mathcal{N}(-0.1, 0.5^2)$$

The episode progresses for at most 40 Mujoco time-steps. The simulation ends if the ball stagnates for 10 consecutive time-steps or does not make progress towards the goal in the y-dimension for 15 consecutive time-steps.

4. RL Platform and Agents

We evaluated three widely used continuous-control reinforcement-learning algorithms under a unified experimental framework. All agents were implemented with the Stable Baselines 3 library (Raffin et al., 2021), ensuring consistent game state. Each policy employed the same multilayer-perceptron (MLP) backbone: seven fully connected layers of 3 000 hidden units. Adam optimizers with identical learning-rate schedules were applied across algorithms, and gradient clipping was enabled to stabilize updates. Below we provide a quick review of the features of each method.

4.1. Proximal Policy Optimization (PPO) (Schulman et al., 2017)

Proximal Policy Optimization is an on-policy, actor–critic algorithm designed to balance between sample efficiency and implementation simplicity. During each update, PPO maximizes the clipped objective:

$$L_{\text{clip}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t\big)\Big],$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

which limits the magnitude of policy updates and preventing destructive oscillations.

4.2. Soft Actor–Critic (SAC) (Haarnoja et al., 2019)

Soft Actor–Critic is an off-policy algorithm that augments the conventional actor–critic framework with an entropy-regularization term, encouraging the policy to remain stochastic and explore the action space:

$$J_\pi = \mathbb{E}_{s\sim\mathcal{D}}\big[\mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q(s,a) - \alpha \log \pi(a \mid s)\big]\big]$$

Two independent critics are trained to minimize a squared-Bellman loss, while a temperature parameter α is tuned automatically to match a target entropy.

4.3. Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018)

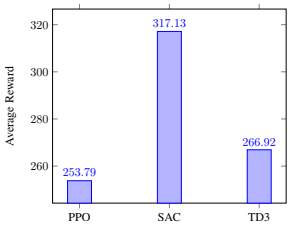
TD3 extends Deep Deterministic Policy Gradient by mitigating overestimation bias through three modifications: (i) maintaining twin critic networks and taking the minimum of their target values, (ii) delaying actor updates with respect to critic updates, and (iii) adding clipped Gaussian noise to the target action.

5. Training Protocol

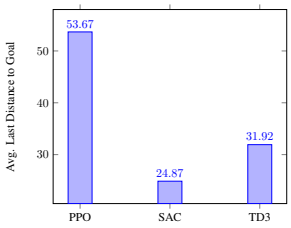
All three algorithms were trained in the foosball environment introduced in Section 3. Each run comprised 15 epochs, where one epoch equaled 10^5 simulation time steps. A replay buffer of capacity 10^6 was pre-allocated and cleared only between independent seeds. To track out-of-sample performance, we performed deterministic evaluations every 3 000 steps, averaging returns over ten held-out episodes with exploration disabled. Hyperparameters not listed above—such as learning rates (3×10^{-4} for actors and critics), discount factor ($\gamma = 0.99$), and gradient norms—were held constant across agents. All experiments executed on a single NVIDIA RTX 4090 GPU.

6. Preliminary Results

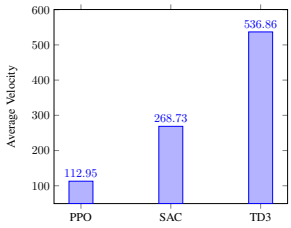
Each trained controller was subjected to a test phase comprising 10^3 independent episodes in the same simulation environment, with exploration noise disabled so that actions were chosen deterministically. For every episode we logged the full trajectory of states, actions, and rewards, and afterward computed the following diagnostics:



(a) Average Reward



(b) Avg. Last Distance to Goal



(c) Average Velocity

Figure 2: Evaluation metrics computed over 10^3 deterministic episodes.

- **Score Rate**: the proportion of episodes in which the ball ultimately crossed the opponent's goal line.
- **Average Distance to Goal**: the mean Euclidean distance between the ball and the goal center, aggregated over *all* time steps in all evaluation episodes.
- **Average Velocity**: the mean speed of the ball, again averaged over every time step across the evaluation set.
- **Average Reward**: the mean cumulative reward obtained per episode under the deterministic policy.
- **Average Last Distance to Goal**: the ball-to-goal distance measured at the *final* time step of each episode, then averaged over episodes.

Overall performance. Under the chosen training horizon and domain-randomization settings, **SAC emerged as the top-performing algorithm**. As depicted in Figure 2, SAC achieved a 42.9% Score Rate, more than doubling the values recorded for PPO and TD3. Correspondingly, its Average Reward was noticeably higher, reflecting the reward bonus associated with a successful goal.

Proximity versus conversion. Although SAC converted the largest share of episodes into goals, Fig. 2b reveals that TD3 ended its episodes almost as close to the target. The Average Last Distance to Goal of TD3 was 31.92 versus 24.87 for SAC. This indicates that TD3 learned to advance the ball into shooting range but lacked a reliable finishing tactic. Given the relatively short 15-epoch budget, it is likely that additional training would allow TD3 to narrow, or even close, the scoring gap.

Ball-handling dynamics. Interestingly, the agents discovered distinct ball-speed regimes despite the absence of an explicit velocity term in the objective. TD3 propelled the ball at roughly twice the speed attained by SAC, which itself moved the ball about twice as fast as PPO. These differences are reflected in the *Average Velocity* metric and suggest that each algorithm gravitated toward a unique trade-off between control precision and shot power.

In summary, the evaluation confirms that SAC currently offers the most reliable end-to-end policy under our training constraints, while TD3 shows promise in ball advancement that could translate into higher scoring with extended training. PPO, although stable, lags behind both in conversion efficiency and ball-movement aggressiveness.

7. Open-Source Benchmark

Foosball offers a compact and compelling testbed for deep reinforcement learning research. Notably, the game's core aspects, including high-speed ball and rod interactions, mechanical contact dynamics, and adversarial gameplay, require a mixture of fine low-level motor control and high-level policy learning and opponent modeling. These characteristics allow for meaningful evaluation of an agent's ability to generalize across domains without being overwhelmed by noise or large mechanical movements. Unlike some RL domains that operate in high-dimensional, unpredictable environments, foosball provides a controlled setting with complex yet well-defined dynamics. Furthermore, the need for precise coordination across multiple rods, and rapid decision-making under adversarial pressure make it a rich test case for learning across various RL frameworks. To support continued research in the foosball domain, we present our custom foosball environment as an open-source physically accurate testing ground for RL algorithms.

8. Conclusion

This paper introduces an end-to-end deep RL pipeline enabling AI agents to infer the physics in a competitive foosball setting, bridging both simulation and physical environments. The pipeline employs a high-fidelity simulation model reproducing the real-world dynamics so that the AI models can implicitly learn and understand dynamics in the foosball game through visual inputs. We believe, by supporting reproducible deployment pipelines and standardized evaluation protocols, the benchmark allows agents trained in simulation to be tested on the real-world foosball table. A leaderboard infrastructure tracks agent performance across domains, enabling fair comparisons and long-term progress tracking. Foosball's structured yet dynamic gameplay presents a well-suited environment for studying sim-to-real generalization, physical scene understanding, competi-

tive RL strategies, and control robustness.

References

Croenen, J., Bürger, J., and Meert, W. Reinforcement learning of action sequences in table football. URL https://api.semanticscholar.org/CorpusID:276604268.

De Blasi, S., Klöser, S., Müller, A., Reuben, R., Sturm, F., and Zerrer, T. Kicker: An industrial drive and control foosball system automated with deep reinforcement learning, 04 2020.

D'Ambrosio, D., Jaitly, N., Sindhwani, V., Oslund, K., Xu, P., Lazic, N., Shankar, A., Ding, T., Abelian, J., Coumans, E., Kouretas, G., Nguyen, T., Boyd, J., Iscen, A., Mahjourian, R., Vanhoucke, V., Bewley, A., Kuang, Y., Ahn, M., Jain, D., Kataoka, S., Cortes, O., Sermanet, P., Lynch, C., Sanketi, P., Choromanski, K., Gao, W., Kangaspunta, J., Reymann, K., Vesom, G., Moore, S., Singh, A., Abeyruwan, S., and Graesser, L. Robotic table tennis: A case study into a high speed learning system. In *Robotics: Science and Systems XIX*, RSS2023. Robotics: Science and Systems Foundation, July 2023. doi: 10.15607/rss.2023.xix.006. URL http://dx.doi.org/10.15607/RSS.2023.XIX.006.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477.

Gashi, A., Hergenröther, E., and Grieser, G. Efficient training of foosball agents using multi-agent competition. In Arai, K. (ed.), *Intelligent Computing*, pp. 472–492, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-37717-4.

Gutierrez-Franco, J., Inlow, J., Graham, J., and Huang, L. Automated foosball table year 1: Final project report. 12 2013.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications, 2019. URL https://arxiv.org/abs/1812.05905.

Hagens, D., Knaup, J. M., Hergenröther, E., and Weinmann, A. Cnn-based game state detection for a foosball table, 2024. URL https://arxiv.org/abs/2404.05357.

Hernández, N., Rebolledo, J., Torres, J., Carvajal, G., and Vargas, F. Design of an experimental platform for the automation of the goalkeeper of a foosball table. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1–7, 2019. doi: 10.1109/CHILECON47746.2019.8988113.

Horst, R., Hagens, D., Hergenröther, E., and Weinmann, A. Real time state detection of a foosball game using cnn-based computer vision. In Arai, K. (ed.), *Intelligent Computing*, pp. 343–354, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-62281-6.

Janssen, R., Best, J., and Molengraft, M. Real-time ball tracking in a semi-automated foosball table. pp. 128–139, 06 2009. ISBN 978-3-642-11875-3. doi: 10.1007/978-3-642-11876-0_12.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. doi: 10.1177/0278364913495721. URL https://doi.org/10.1177/0278364913495721.

Kurach, K., Raichuk, A., Stańczyk, P., Zajac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., and Gelly, S. Google research football: A novel reinforcement learning environment, 2020. URL https://arxiv.org/abs/1907.11180.

Margolis, G. B., Yang, G., Paigwar, K., Chen, T., and Agrawal, P. Rapid locomotion via reinforcement learning, 2022. URL https://arxiv.org/abs/2205.02824.

Moos, J., Derstroff, C., Schröder, N., and Clever, D. Learning to play foosball: System and baselines. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4326–4332. IEEE, May 2024. doi: 10.1109/icra57147.2024.10611321. URL http://dx.doi.org/10.1109/ICRA57147.2024.10611321.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017a. URL http://arxiv.org/abs/1710.06537.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2017b. URL https://api.semanticscholar.org/CorpusID:3707478.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, 2017. URL http://arxiv.org/abs/1703.02702.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Rohrer, T., Samuel, L., Gashi, A., Grieser, G., and Hergenröther, E. Foosball table goalkeeper automation using reinforcement learning. In *Lernen, Wissen, Daten, Analysen*, 2021. URL https://api.semanticscholar.org/CorpusID:240290331.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Tammewar, A., Chaudhari, N., Saini, B., Venkatesh, D., Dharahas, G., Vora, D., Patil, S., Kotecha, K., and Alfarhood, S. Improving the performance of autonomous driving through deep reinforcement learning. *Sustainability*, 15(18), 2023. ISSN 2071-1050. doi: 10.3390/su151813799. URL https://www.mdpi.com/2071-1050/15/18/13799.

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., and Stone, P. Deep reinforcement learning for robotics: A survey of real-world successes, 2024. URL https://arxiv.org/abs/2408.03539.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. URL https://api.semanticscholar.org/CorpusID:2413610.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Weigel, T. Kiro - a table soccer robot ready for the market. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4266–4271, 2005. doi: 10.1109/ROBOT.2005.1570776.

Weigel, T. and Nebel, B. Kiro – an autonomous table soccer player. In Kaminka, G. A., Lima, P. U., and Rojas, R. (eds.), *RoboCup 2002: Robot Soccer World Cup VI*, pp. 384–392, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45135-8.

Zhang, D. and Nebel, B. Learning a table soccer robot a new action sequence by observing and imitating. pp. 61–, 01 2007.