Reinforcement Learning for Foosball in a Custom MuJoCo Environment

Patarada Yontrarak (ppy2104), Kai Zhang (kz2560), Victoria Gonzalez (vcg2110)

***Abstract*—Foosball is a fast, contact-rich tabletop game that demands rapid perception, planning, and continuous control under interaction. Rather than assuming a well-calibrated simulator, this work focuses on environment design and validation in MuJoCo. We collect real-world foosball ball trajectory data and use it to guide simulator tuning, including ball dynamics, contact behavior, and reset logic. To evaluate whether the resulting environment exposes meaningful structures and results, we train Soft Actor–Critic (SAC) and Truncated Quantile Critics (TQC) under identical conditions. By analyzing the learning curves, episode statistics, and goal-related metrics, we are able to see that the redesigned environment avoids degenerate dynamics and supports stable learning, while also revealing evaluation pitfalls such as metric inflation due to reset bias. These results highlight the importance of careful simulator validation before interpreting RL performance in contact-rich tasks.**

I. Introduction

Modern robot learning often assumes a high-quality simulator is already available, but in practice, building a useful simulation of a contact-rich task is itself a non-trivial robotics problem. Foosball is a good example: the task involves fast ball dynamics, intermittent contacts with slender bodies, and multi-agent coordination, yet most RL work on table-top games focuses on abstract board representations rather than physically realistic play.

A foosball table is also a potentially cheap, safe, and repeatable platform for studying coordination and control. However, before we can explore sophisticated multi-rod policies, we need a well-calibrated multi-rod environment that (1) produces realistic contacts and ball motion and (2) is learnable by standard continuous-control RL algorithms.

In this project, we focus on the environment-building and calibration problem. We collected motion data from a real foosball table in Lerner Hall (ball trajectories, rod positions, contact events), and used it to design and tune a minimal MuJoCo environment: table geometry, collision bodies, initializations to mimic real physics as much as possible. We then used Soft Actor–Critic (SAC) and Truncated Quantile Critics (TQC) as probes to test whether our environment and reward shaping supported meaningful ball–rod interactions beyond what a random or scripted controller could achieve.

Our project had three objectives:

1) **Data-informed environment design:** Build a MuJoCo foosball environment whose geometry, ball height, and player contact behavior are consistent with real-table data, including gravity, friction, ball–ground and ball–rod contacts, typical ball speeds, and drift.
2) **Task and reward formulation:** Define a foosball playing task and associated reward function that balance ball progress toward the goal, contact quality, and time or control penalties. We evaluate multiple reward parameterizations for each policy, as described in Section III.
3) **RL-based evaluation of environment quality:** Train Soft Actor–Critic (SAC) and Truncated Quantile Critics (TQC) on the same environment and compare their learning behavior using goal-related metrics, assessing performance under the different reward configurations introduced in (2).

Our main contributions are:

1) A data-calibrated MuJoCo foosball environment where contact behavior (ball–rod, ball–ground, walls) is grounded in observations from a physical table, rather than purely hand-tuned.
2) A task and reward design for a multi-rod shooting problem, together with evaluation metrics (goal rate, contact counts, ball progress) that reveal when the environment is mis-specified or too easy/hard.
3) An empirical study using SAC and TQC using different reward logic design, highlighting the sensitivity of RL performance to environment design.

II. Related Work

Foosball sits at the intersection of contact-rich tabletop manipulation and multi-agent continuous control. Prior work relevant to our project falls into three threads: (i) foosball-specific simulators and benchmarks, (ii) techniques for reliable contact simulation and sim-to-real calibration, and (iii) modern off-policy RL methods commonly used as "learning probes" in robotics. Our work connects these threads by treating environment fidelity and evaluation design as first-class problems, rather than assuming the simulator is already trustworthy.

A. Physically Grounded Foosball Environments and Benchmarks

Recent efforts have introduced MuJoCo-based foosball environments to study learning under realistic rigid-body dynamics and intermittent contacts [3, 4]. These simulators typically provide rod kinematics, player geometries, and a rolling ball with collisions, enabling end-to-end learning with continuous actions. Such benchmarks are valuable because they move beyond abstract game-state representations and expose the control challenges created by fast ball motion and thin-body contacts.

At the same time, foosball benchmarks often inherit a practical assumption shared by many robotics simulators: that a reasonable set of default contact parameters and joint

constraints will yield physically plausible rollouts. In our own early iterations, small modeling choices (e.g., ball height and joint type, contact solver tolerances, friction/damping) produced degenerate regimes such as "frozen" ball motion or constraint-dominated behavior, where learning curves become hard to interpret. These failure modes motivate our focus: rather than treating the simulator as fixed, we use real ball trajectories to guide environment corrections and to validate that the task dynamics are not artifacts of mis-specified contacts.

B. Contact-Rich Simulation and Data-Driven Calibration

Accurately simulating contact-rich interactions remains a known challenge in robotics, where small errors in geometry, friction, restitution, damping, or solver settings can lead to qualitatively wrong outcomes [5]. A common response is to incorporate data: simulation parameters are tuned using real measurements, or training is made robust to mismatch via domain randomization. In tabletop settings, even simple rolling/sliding behaviors can be sensitive to friction and contact modeling, making calibration particularly important when performance is evaluated via long-horizon learning.

Our approach is aligned with this line of work but targets a specific and practical calibration objective: ensuring that basic ball dynamics (rolling, slowing, bouncing) and contact events are plausible enough that downstream RL training is meaningful. We do not attempt full system identification from monocular video; instead, we use real trajectories as constraints to detect and correct concrete failure modes (e.g., penetration/constraint explosions, ball stagnation) and then treat RL behavior as a secondary diagnostic of whether the environment exposes learnable structure rather than bugs.

C. Off-Policy RL as a Diagnostic for Environment Quality

Off-policy actor–critic methods are widely used for continuous-control robotics because they are sample-efficient and can learn from replay buffers [1]. Distributional variants such as TQC further aim to mitigate overestimation bias by truncating high quantiles in target computation [2]. In many papers these algorithms are the main contribution and are evaluated primarily on final return.

In this project, SAC and TQC play a different role. We use them as "probes" to evaluate whether the environment, resets, and rewards produce interpretable learning signals. In particular, our results show that goal-based metrics can be inflated by reset bias (e.g., a favorable initial serve leading to fast terminations), so algorithmic success alone is insufficient evidence of true contact-rich control. This perspective is consistent with broader lessons from robotics benchmarking: reliable conclusions about policies require careful task design, well-specified evaluation protocols, and diagnostics that separate genuine control from simulator artifacts.

D. Gap Addressed by This Work

Taken together, prior foosball benchmarks provide the right high-level structure for studying the game [3, 4], and prior calibration work highlights the sensitivity of contact dynamics to modeling error [5]. However, there remains a gap in practical procedures for validating that foosball ball–table and ball–rod dynamics are realistic enough that RL results reflect skill rather than simulator quirks. Our contribution is a trajectory-informed validation/tuning loop and an evaluation analysis (via SAC/TQC) that explicitly surfaces reset-induced metric inflation, providing a more cautious and reproducible basis for interpreting learning outcomes in contact-rich foosball simulation.

III. Method & Approach

Our method follows an end-to-end pipeline. First, real foosball videos are processed using an OpenCV-based tracker to extract ball trajectories. Next, homography-based mapping to metric coordinates informed simulator design. The MuJoCo environment was tuned using trajectory structure and qualitative dynamics. Finally, the environment was evaluated with off-policy continuous-control agents, specifically SAC and TQC.

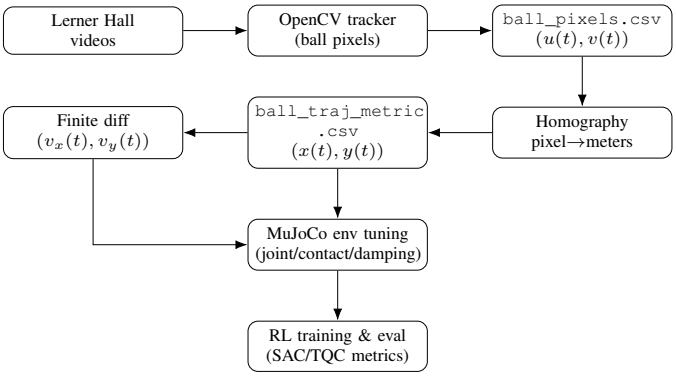

Fig. 1: Data-to-simulation evaluation pipeline

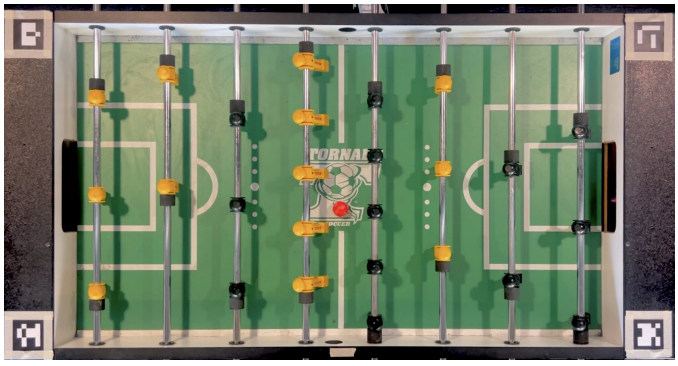

Fig. 2: 4 ArUco tags for table corner detection and alignment

A. Real-World Data Collection & Extraction Pipeline

A major portion of our work was building a trustworthy pipeline from real footage to quantities/physics that can constrain simulator design.

1) Real-World Foosball Recordings: We record overhead videos of the Lerner Hall foosball table using 3 balls separately (red, purple, and blue) to improve visual contrast and tracking reliability. We also recorded gameplay at three pace bands, slow, moderate, and fast—to capture trajectories across a range of difficulties and velocities.

2) Ball detection in video (pixel domain): We run an OpenCV tracker configured for high-contrast ball detection derived from [4, 6]. As shown in Figure 3, the green bounding box is tracking the red foosball. For each recording, the tracker outputs a CSV file containing frame timestamps and pixel coordinates $(u(t), v(t))$ of the ball. Figure 1 summarizes the full pipeline. Four ArUco tags were stuck on the four corners of the foosball table. These tags are detected by the OpenCV model to locate the position of the table as shown in Figure 2.

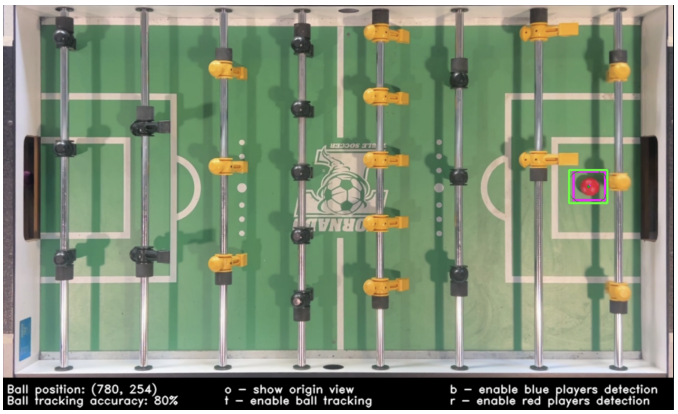

Fig. 3: OpenCV ball tracking on Lerner Hall footage (ball bounded by green box)

3) Pixel-to-meters mapping via homography: To convert detections into physically meaningful positions, we fit a planar homography using four corner correspondences between the image plane and known table coordinates:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.$$

Applying H to each detection yields metric ball positions $(x(t), y(t))$ on the table plane. This step is crucial: it removes camera perspective distortion and makes trajectories comparable to simulator coordinates and distances.

4) Velocity estimation and trajectory products: We estimate planar velocity via finite differences on the resampled metric trajectory:

$$v_x(t) \approx \frac{x(t+\Delta t) - x(t-\Delta t)}{2\Delta t},$$

$$v_y(t) \approx \frac{y(t+\Delta t) - y(t-\Delta t)}{2\Delta t}.$$

B. Pipeline Refinement: Camera Calibration

While figuring out what would go into our final pipeline, we attempted to calibrate the camera with OpenCV. We used a checkerboard with 9×6 inner squares and 25mm sized squares, taking images across varied pitch/roll/yaw.

(a) Checkerboard lying flat on the table.

(b) Checkerboard viewed at an oblique angle.

Fig. 4: Camera calibration images used for monocular calibration.

OpenCV calibration with 10 out of 19 accepted views produced an RMS reprojection error of 4.561884 px and a mean reprojection error of 0.593244 px. The resulting intrinsics and distortion coefficients were:

$$K \approx \begin{bmatrix} 2665.605 & 0 & 1577.011 \\ 0 & 2693.381 & 2260.714 \\ 0 & 0 & 1 \end{bmatrix}$$

$$d = (k_1, k_2, p_1, p_2, k_3) =$$

$$(0.2269, -0.8207, 0.01662, -0.0056, 0.7033).$$

Here $f_x \approx 2665.6$ px, $f_y \approx 2693.4$ px, and the principal point $(c_x, c_y) \approx (1577.0, 2260.7)$ px.

However, we realized that the recording device applied internal calibration by default, and additional calibration did not yield reliable improvements. Thus, it was removed from our pipeline.

C. MuJoCo Environment Design

Our simulation environment is implemented in MuJoCo and is derived from the RoboFoosball benchmark [3]. The environment models a regulation foosball table, a rolling ball, and multiple rods with translational and rotational degrees of freedom.

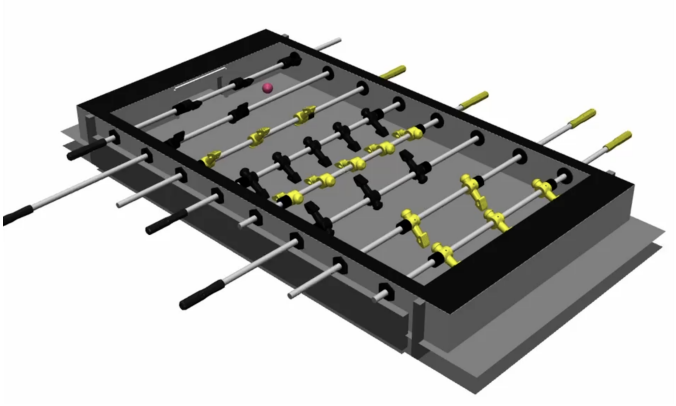

Fig. 5: Our MuJoCo Simulation Environment

1) Action space: This environment exposes the full 8-dimensional action vector for both the protagonist and antagonist,

$$a \in [-20, 20]^8,$$

with two controls (linear slide and rotation) per rod (goal, defense, midfield, attack).

2) Observation space: The observation is 36-dimensional and concatenates planar ball state and all rod joint states for both players:

$$\underbrace{(x, y, \dot{x}, \dot{y})}_{\text{planar ball state}} + \sum_{\substack{\text{2 players} \\ \text{4 rods}}} \Big[\underbrace{s}_{\text{slide}}, \underbrace{\dot{s}}_{\text{slide vel}}, \underbrace{r}_{\text{rotation}}, \underbrace{\dot{r}}_{\text{rot vel}} \Big]$$

$$= 4 + 8 \times 4 = 36.$$

3) Reset and Dynamics: At reset, the ball's planar position is set through:

- **Ball joint:** model the ball as a single free joint `ball_free` with (x, y, z) translation and associated velocities.
- **Initial pose:** place it at the table center in (x, y) at a fixed height $z = 0.08$, matching the XML model.
- **Initial velocity:** sample a random planar "serve" in the $+y$ direction (with small lateral noise) so each episode starts with an active rally.
- **Physics parameters:** add modest damping on the translational DOFs and set gravity to $[0, 0, -9.81]$ for realistic rolling behavior.

4) Reward: Let y_t be the current ball y-position and y_{t-1} the previous one. The protagonist always attacks in the $+y$ direction. The new reward is

$$R_t = R_{\text{progress}} + R_{\text{dist}} + R_{\text{goal}} + R_{\text{own}} - R_{\text{ctrl}},$$

where:

- **Progress term:** we compute forward progress

  $$\Delta y = \max(y_t - y_{t-1}, 0), \qquad R_{\text{progress}} = 50\,\Delta y,$$

  rewarding only positive movement toward the attacking goal.

- **Distance shaping:** we introduce a virtual goal y_{virtual} at a fraction of the table length

  $$d = |y_{\text{virtual}} - y_t|, \qquad R_{\text{dist}} = \frac{5}{1 + d},$$

- **Goal bonus/ own-goal penalty:** when $|y_t|$ crosses the physical goal lines, we give

  $$R_{\text{goal}} = \begin{cases} 1000, & \text{protagonist scores in } +y \\ 0, & \text{otherwise,} \end{cases}$$

  $$R_{\text{own}} = \begin{cases} -1000, & \text{ball crosses } -y \text{ goal (own goal)} \\ 0, & \text{otherwise.} \end{cases}$$

5) Termination: Any episode terminates if any of the following is true:

- **Goal scored:** the ball crosses either goal line (same thresholds used in the reward).
- **Ball stagnant:** the ball's speed $\|\dot{\mathbf{x}}_t\|$ remains below a small threshold for a fixed number of consecutive steps (ball effectively stopped).
- **Max episode time:** the simulated time exceeds a fixed horizon, using `simulation_time`.

D. RL Algorithms Used: SAC and TQC

We train two off-policy continuous-control RL algorithms, Soft Actor–Critic (SAC) and Truncated Quantile Critics (TQC), as evaluation tools for our environment fidelity and reward learnability, rather than as the primary contribution.

a) Soft Actor–Critic (SAC): SAC is an off-policy actor–critic method for continuous actions that optimizes a maximum-entropy objective. SAC learns a stochastic policy $\pi_\theta(a \mid s)$ and critic(s) $Q_\phi(s, a)$ by maximizing expected return while encouraging exploration.

b) Truncated Quantile Critics (TQC): TQC is a distributional extension of SAC that represents the return distribution via quantile regression and reduces overestimation bias by truncating the highest quantiles when forming targets.

IV. Experiments & Results

This section reports quantitative and qualitative evaluation of the custom MuJoCo foosball environment using SAC and TQC, including learning curves, episode statistics, and goal-related metrics.

A. Experimental Setup Overview

Our evaluation is structured around two linked questions: (i) does the simulator reproduce usable, repeatable ball dynamics consistent with real table behavior, and (ii) do standard continuous-control RL methods produce learning signals that reflect meaningful control rather than simulator artifacts?

1) Datasets & inputs: Our dataset is a set of Lerner Hall foosball videos collected under controlled conditions (single-ball rollouts and gameplay at multiple pace bands). To improve detection effectiveness, we used multiple ball colors (red/purple/blue) to maximize contrast against the table surface with varying play strategies such as slowing down the play speed. The raw videos are processed into time-stamped ball detections and then converted to metric trajectories used for calibration/validation.

2) Environment: We train and evaluate in our custom MuJoCo foosball environment derived from an existing CAD model explained in Section III.C [3]. The key evaluation idea is that if the physics are mimicking the real world physics as much as possible, then learning curves and episode statistics should be interpretable and stable across runs. If the physics are broken, both SAC and TQC collapse to trivial behavior.

3) Compute & software: Tracking is done with an OpenCV-based pipeline [4, 6]. RL training uses Stable-Baselines3 SAC and sb3_contrib TQC in the same environment wrapper explained in Section III.C, changing only the algorithm.

B. How Real Data Informed the Simulation Setup

We use the extracted trajectories primarily as constraints on simulator behavior: the ball should roll smoothly on the plane, slow down under friction/damping, and avoid pathological penetration/constraint explosions. Concretely, the data guided:

- Ball state representation: modeling the ball with a physically appropriate joint configuration (free motion rather than constrained planar joints that can introduce solver artifacts).
- Ball pose initialization: placing the ball at a fixed, realistic height above the table and serving with small planar velocity consistent with observed rollouts (avoid "instant score" resets when the ball is already effectively in the goal region).
- Contact plausibility checks: using trajectory smoothness and speed decay as sanity checks that contact settings do not freeze the ball or inject energy unnaturally (a failure mode we saw in the baseline).

However, we also recognized some limitations even with this pipeline. Calibration is not fully identified from monocular trajectories alone: multiple friction/damping/contact parameter combinations can produce visually similar rollouts. As a result, we treat RL training as a secondary probe: if the task is too easy/hard due to physics, training curves and episode statistics will reveal it (e.g., episodes ending instantly, or no progress across millions of steps).

C. Metrics and Baselines

We track training-time scalars (exported from our runs as CSV) for both algorithms:

- Episodic return (and a rolling mean over the last 50 episodes, denoted Return_{50}).
- Episode length (rolling mean EpLen_{50}), which is highly diagnostic in our environment because many terminations correspond to either (i) goal events or (ii) stagnation/timeouts.
- Goal-related rates smoothed over 50 episodes: WinRate_{50}, LoseRate_{50} (own goals), and TermGoal_{50} (fraction of episodes terminating due to a goal event).

Our primary baseline is algorithmic: SAC vs. TQC on the same environment and reward. (A full control-policy baseline like "do nothing" or scripted rods is straightforward to add, but was not logged in the attached training exports.)

D. RL Training

We trained SAC and TQC under identical environment wrapper, reward computation, termination logic, and comparable training budget (on the order of ≈ 0.8 million environment steps for the exported runs). We used the same MLP-style policy/value networks and the same replay-buffer-based off-policy training loop; only the learning algorithm class differs (SAC vs. TQC).

E. Results: Learning Curves and Episode Statistics

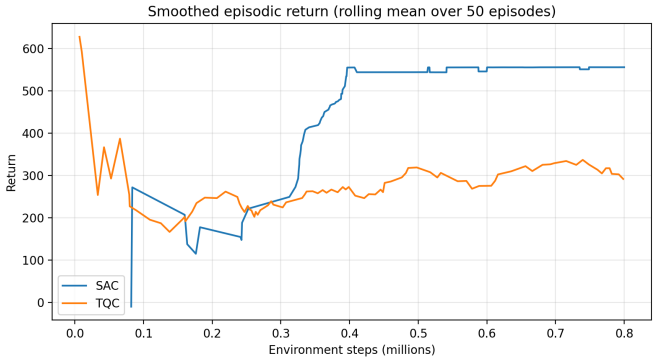

Fig. 6: Smoothed episodic return Return_{50}

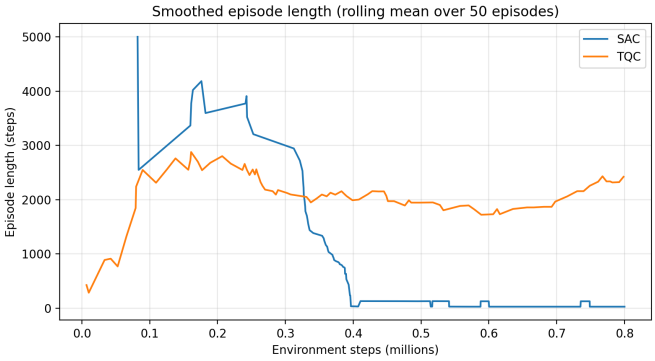

Fig. 7: Smoothed episode length EpLen_{50}

Figures 6 and 7 show that SAC converges to a stable Return_{50} plateau, while TQC reaches higher peaks but is less stable. This is consistent with a common pattern in contact-rich

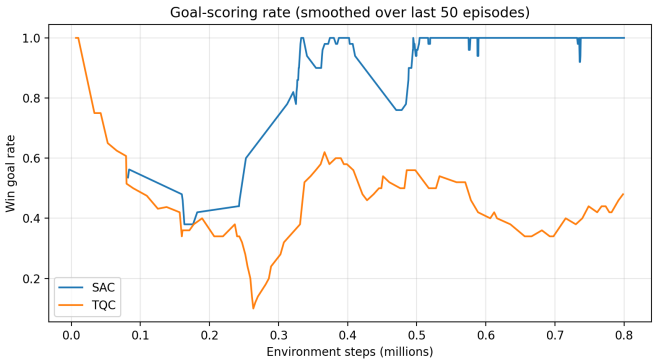

Fig. 8: WinRate_{50} goal scoring rate

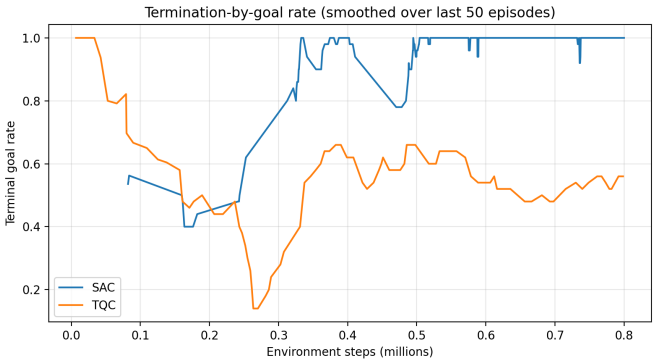

Fig. 9: TermGoal_{50} termination by goal

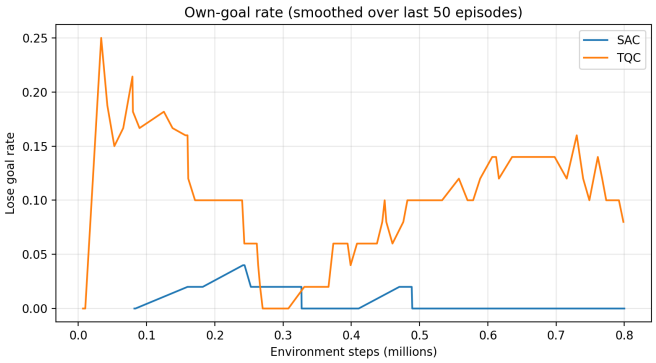

Fig. 10: LoseRate_{50} own goals

tasks: distributional critics can discover aggressive strategies that occasionally perform very well, but can also be more sensitive to reward/termination quirks.

However, episode length reveals a critical nuance: SAC's EpLen_{50} collapses to very short episodes late in training (tens of steps), while TQC maintains much longer episodes (often thousands of steps). In our environment, this typically indicates that SAC episodes are terminating quickly due to goal events (or a termination condition firing quickly), whereas TQC spends more time in extended rallies or timeouts.

Goal-rate curves in Figures 8, 9, and 10 show much higher WinRate_{50} for SAC than TQC, with near-zero LoseRate_{50} for both. Combined with the episode-length collapse, this suggests that SAC is frequently reaching the goal termination condition quickly. This is encouraging insofar as it shows the environment is no longer degenerate (the ball moves and goal events occur), but it also exposes a likely evaluation inflation risk: if the reset "serve" already biases the ball to roll into the $+y$ goal region, then high win-rate does not necessarily imply that the rod policy learned meaningful interception or striking behavior.

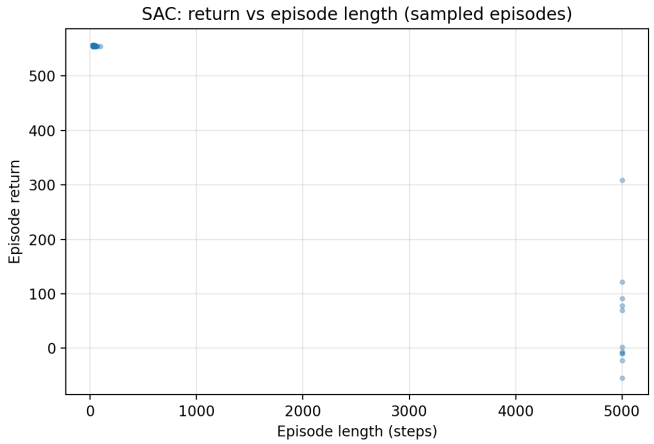

Fig. 11: SAC return vs episode length

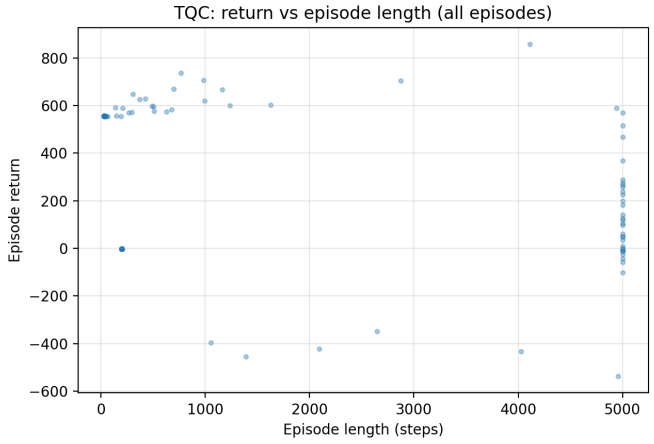

Fig. 12: TQC return vs episode length

To make this interaction explicit, Figure 11 plots return vs. episode length for SAC. The strong cluster of short episodes with high return is consistent with fast terminations being rewarded, which may be dominated by environment initialization rather than contact-rich control.

F. Summary Table

Table I aggregates final and best metrics from the exported runs (smoothing windows as defined above). We report these numbers to summarize trends, but we interpret them cautiously due to the reset/termination sensitivity discussed next.

TABLE I: Summary of exported training metrics with 50 episode smoothing where applicable

Algo	Steps M	Ret_{50} final	Ret_{50} best	Len_{50} final	Win_{50} final	Goal_{50} final
SAC	0.80	556.00	556.03	30.02	1.00	1.00
TQC	0.80	292.03	627.95	2422.24	0.48	0.56

G. Critical Analysis, Limitations, and Improvements

1) What the results suggest: The strongest positive takeaway is that after the environment redesign, learning curves are no longer flatlined and goal events occur reliably. SAC shows stable learning and converges to consistent performance under our current metrics; TQC occasionally achieves higher return but with instability, which matches our qualitative experience that distributional methods can be more sensitive to reward/termination choices.

2) Main limitation (likely metric inflation): The combination of (i) extremely high WinRate_{50} for SAC, (ii) very short EpLen_{50} late in training, and (iii) high TermGoal_{50} indicates that many episodes may be terminating quickly due to the goal condition. This can happen even without meaningful rod control if the reset "serve" imparts forward velocity that carries the ball into the goal region. In this case, return and win-rate overestimate policy quality.

3) Unexpected finding: TQC produces significantly longer episodes than SAC at similar step budgets. This may reflect that TQC learns behaviors that keep the ball in play longer (delaying termination), or that it fails to convert trajectories into goals as reliably. In a more realistic setting (with an opponent and harder resets), longer controlled rallies could be desirable; under the current termination structure, they can also correspond to timeouts/stalls.

4) Next improvements: To make evaluation more faithful to "foosball skill" (contact-rich control), we would:

- Harden resets: randomize ball position away from the goal and reduce/reset initial velocity so scoring requires at least one meaningful contact.
- Add contact-conditioned metrics: log rod–ball contact count/impulse and measure "progress after contact" to distinguish passive rolling-in from active striking.
- Add an opponent or scripted defender: prevent trivial straight-line scoring and turn goal-rate into a meaningful skill measure.
- Run multiple seeds and fixed evaluation rollouts: reduce variance and ensure reported improvements generalize beyond a single training trace.

V. Conclusion

We studied foosball RL primarily through the lens of simulator design and validation. Our first objective 1 was met by using real Lerner Hall ball trajectories to guide ball dynamics, contact behavior, and reset logic, eliminating the degenerate "frozen ball" failure mode observed in the baseline simulator. Our second objective 2 was met by implementing a goal-driven reward with shaping terms and clear termination conditions to produce interpretable episodes. For the third objective 3 we trained SAC and TQC under identical conditions and observed stable learning signals once the environment was corrected: SAC achieved consistently high goal rates but with very short episodes, while TQC produced longer rallies with lower goal conversion. This revealed a key lesson: goal-based metrics can be inflated by reset bias, so episode statistics and diagnostics beyond return are necessary to interpret performance. Future work will harden resets, add contact-conditioned metrics, and introduce an opponent (scripted or learned) to make goal rate a more faithful measure of foosball skill.

VI. Team Member Contribution Statement

Algorithm choosing: Patarada Yontrarak led the selection and configuration of reinforcement learning algorithms (SAC and TQC) and defined evaluation metrics.

Coding/implementation: Kai Zhang implemented the MuJoCo environment modifications and RL training pipeline. Patarada Yontrarak contributed to the data extraction pipeline and camera calibration.

Data collection and preprocessing: Patarada Yontrarak, Kai Zhang, and Victoria Gonzalez equally contributed to this section.

Experiment design: Kai Zhang designed experimental protocols, training conditions, and comparison methodology.

Writing responsibilities: Patarada Yontrarak, Kai Zhang, and Victoria Gonzalez equally contributed to this section.

System integration: Kai Zhang led end-to-end system integration.

References

[1] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019. URL https://arxiv.org/abs/1812.05905.

[2] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics, 2020. URL https://arxiv.org/abs/2005.04269.

[3] Matthew So, Kwansoo Lee, Judah Goldfeder, and Hod Lipson. Open-source foosball benchmark for deep reinforcement learning. In *OpenReview*, 2022. https://openreview.net/forum?id=i5HQkbi9UY.

[4] Maciej Trzaskowski. foosball. https://github.com/mtszkw/foosball, 2023.

[5] Toshiaki Tsuji, Yasuhiro Kato, Gokhan Solak, Heng Zhang, Tadej Petrič, Francesco Nori, and Arash Ajoudani. A survey on imitation learning for contact-rich tasks in robotics, 2025. URL https://arxiv.org/abs/2506.13498.

[6] Patarada Yontrarak. foosball_detection. https://github.com/pruyontrarakk/foosball_detection, 2025.