Penalty Shooting using DRL in RoboCup 3D Simulation

João Alves | Marco Rocha | Rúben Monteiro
Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
e-mail: {up202007614 | up202004891 | up202006478}@up.pt

***Abstract*—This paper explores the challenges of using reinforcement learning to train a NAO robot model for goal scoring in the RoboCup 3D Simulation League while having another robot as goalie with heuristic-based behavior. The main objective of this work is to align ourselves with the innovative spirit of RoboCup and to gain a deeper understanding of the complexities associated with reinforcement learning training in the hope of further contributing to the development and refinement of intelligent humanoid football-playing robots.**

***Index Terms*—Simulation Football, RoboCup 3D, Reinforcement Learning, Humanoid Robot, Goal-scoring**

I. Introduction

The RoboCup [1], first established in 1997, is a prominent platform that promotes AI robots by presenting participants with diverse and complex challenges, among them, of interest for this project: football. The RoboCup's 3D Simulation, later introduced in 2004 and presented in Figure 1, has undergone significant advancements, transitioning from utilizing simple spherical robots to humanoid NAO-model robots [2], with 25 degrees of freedom as shown in Figure 2. The competition is hosted yearly throughout the world with the ambitious goal of developing a robot team capable of competing against the world football champion team as of 2050.

Fig. 1. RoboCup 3D Simulation (Source)

This paper studies the use of reinforcement learning to help the robot agent learn optimal behaviors in an environment through trial and error to maximize a reward function. Going into further detail, the intended goal is to create a player capable of walking forward towards the ball placed spawned in its immediate frontal area and then place itself in the best angle to kick the ball in the direction of the goal, trying to score, while keeping into account an heuristic-based goalie robot.

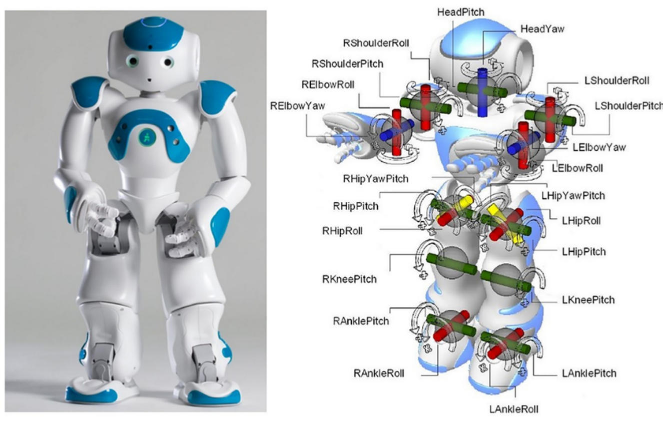

Fig. 2. NAO Programmabl Robot Joints (Source)

This work builds on top of *FC Portugal*'s source code (https://github.com/m-abr/FCPCodebase), improving humanoid robot goal scoring with the already existing Gym environment tools to train our agent. Our code can be found in the following repository https://github.com/m21ark/ri2.

Regarding structure, in subsequent sections this document delves into related work (Section II), details the methodology used (Section III) and the results originated from it (Section IV), concluding with Section V.

II. Related Work

RoboCup is a global scientific initiative aimed at advancing the state of the art in intelligent robotics that has many different domains, with this work's focus being only on the RoboCup 3D Soccer Simulation League.

The team based its work on the *FC Portugal* codebase [3] and used deep reinforcement learning to train an agent capable of performing penalties. Due to limitations in both time and hardware, our work builds on top of previously defined custom behaviors that are included in the codebase.

To perform the training, the team used a Proximal Policy Optimization (PPO) algorithm provided by *OpenAI's Baselines* Python library. The hyperparameters were also adjusted according to the project needs.

III. Methodology

This section uncovers the reinforcement-learning attacker agent training and testing, as well as the heuristic-based goalie's key logical components. The former focuses learning to walk to the ball and kick it towards the goal, while the latter uses rule-based reasoning to prevent scoring. This initial setup is depicted in Figure 3.

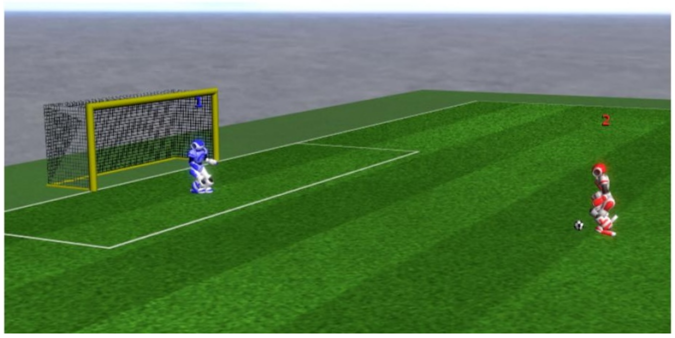

Fig. 3. Starting Position of Penalty Scenario

The goalie, *D*, always spawns in the same position, in the center of the goal, but the attacker spawns randomly in a predefined area near the goal to make it a more challenging and robust training, as seen in Figure 4.

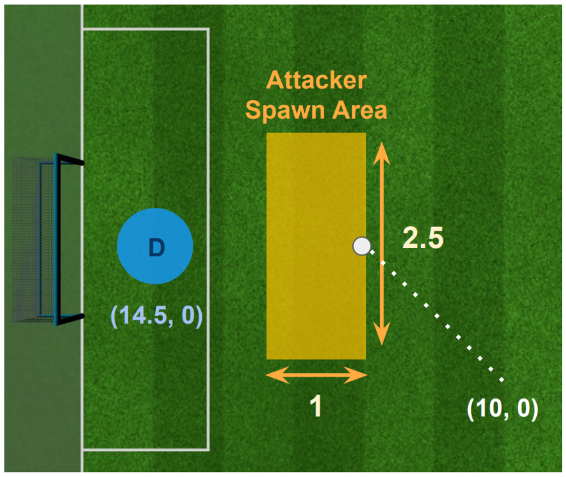

Fig. 4. Top-Down View of the Spawning Areas

A. Heuristic-based Goalie

This subsection describes the implementation of our goalie agent, designed to dynamically respond to incoming ball trajectories and execute different defend actions accordingly. The functionality is encapsulated in a *MyAgentDefender* class that extends the source code's *Base_Agent*.

The agent leverages data such as the ball's position, velocity and direction and uses a state-driven approach, schematized in Figure 5, to make basic defensive decisions. The possible states are as follows:

- **Initialization:** The agent initializes its internal state and waits for a brief period before moving to its default state.
- **Ball Tracking:** This is the default agent state, where the agent moves sideways to place itself between the ball and the goal. If the ball gets too close to the goal, it either moves sideways to block it or transitions to the diving state if the ball speed is too great.
- **Diving to defend:** The agent determines the direction to jump based on the ball's lateral direction, diving either left or right, in parallel to the goal, as exemplified in Figure 6.
- **Diving Recovery:** After diving, the agent executes a *get_up* behavior to recover its upright posture, upon which it resumes the ball tracking state to continue defending.

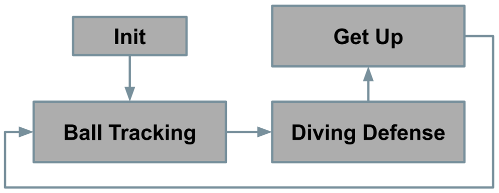

Fig. 5. Goalie State Machine

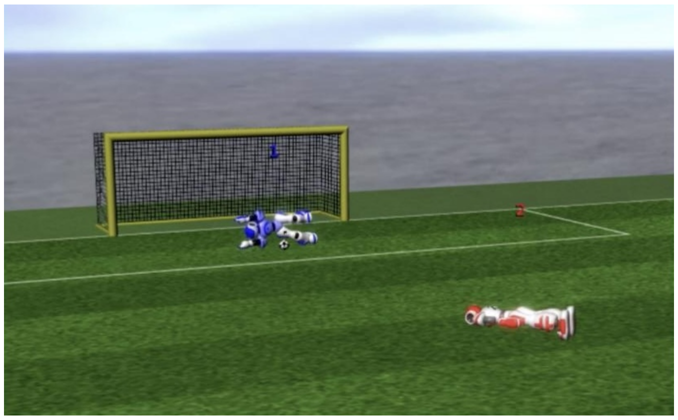

Fig. 6. Successful goalie defense by diving

B. Attacker Agent

The attacker is designed to score penalty goals, and unlike the goalie, it is trained using reinforcement learning methods. The logic of this agent is encapsulated in the *MyPenalty* class that extends the *gym.Env* class.

The main implementation features considered are the *observation space*, the *action space* and the *reward function*. Throughout the work, these were updated multiple times, and their dimensionality had to be significantly reduced when compared to the first approaches to considerably reduce the training time while obtaining better and more consistent results. In this document we only present the best model's configuration.

1) Observation Space: The observation space was comprised of the following parameters:

- **Step Counter (1)** - Value that counts the steps since the beginning of the episodes
- **Last Action (1)** - Value that indicates what the previous action was (0 if *Basic_Kick* or 1 if *Walk*)
- **Player Position (2)** - The current X and Y position values of the attacker
- **Ball Position (2)** - The current X and Y position values of the ball
- **Vector Player/Goal (2)** - A vector in X and Y from the player position to the goal's center

In total, 8 values are used to characterize the observation space of the attacker. Although other information could be and was briefly incorporated, it increased the training time and often it would not converge to better solutions.

2) Action Space: Regarding the action space, the following parameters and respective intervals are used:

- **Behavior (1) [0, 1]** - Thresholded value at 50% that decides whether to perform a kick or walk
- **Kick Direction (1) [-30, 30]** - Value for the degree direction of the kick for a cone shaped kick area
- **Walk Distance (1) [0, 0.5]** - Value that influences walk speed when approaching the final target (0 stops, and 0.5 keeps running when near the target)
- **Walk Orientation (1)** $[-\pi, \pi]$ - Value that indicates the orientation of the torso during the walk
- **Walk Target Position (2) [-3, 3]** - The X and Y position of the target location

In total, 6 values were used for the action space. Limiting their effective range to useful intervals proved to be a crucial step during training, as it greatly increases training performance because the agent does not have to test nonsensical values in the *float32* range.

3) Reward Function: The reward function takes into consideration multiple factors to guide the robot's behavior through the learning process. The team distinguishes two types of rewards: small step rewards to incentivize early action and greater region-based rewards, given at the end of a training episode.

Because we want the agent to perform the penalty kick as early as possible and to avoid many of the incorrect behaviors observed during our experiments, the default reward is set to -1 to incentivize the agent's action and at each simulation step, not kicking the ball results in an additional penalty of -5 points. Furthermore, to discourage the robot from engaging in behaviors other than moving to the ball and kicking it, a reward of +0.5 points is granted per step if the agent is in the vicinity of the ball and a penalty proportional to the player-ball distance is applied if the agent chooses to wander away without performing a kick.

To expedite training, the team decided to allow the attacker only one kick attempt. Consequently, if the attacker falls without kicking the ball, the episode is terminated early with a severe penalty of -2000 points. Falls occurring after the kick are not penalized, as the kick significantly destabilizes the attacker, often resulting in a fall and our attempts to impose a no-falling condition worsened the training process.

Then, there are the terminal regions illustrated in Figure 7, which define the type of reward the agent receives based on the ball's position at the end of a training episode.

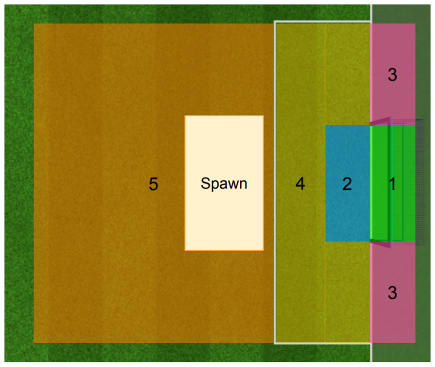

Fig. 7. Different Reward Areas

- **Area 4/5** - These are the outer areas of the field where the attacker spawns and is incentivized to kick the ball by the step rewards mentioned above. If the episode ends with the ball in these areas, a heavy penalty of $-1500 * BallToGoalDistance$ is inflicted.
- **Area 3** - If the ball ends up in this area, it indicates that the shot missed its target. A penalty of -150 points is applied, with a multiplier based on the horizontal offset from the center of the goal.
- **Area 2** - If the ball reaches the area near the goal, +500 points are awarded because the attacker was nearly successful and only the direction or kick force needed slight corrections.
- **Area 1** - If a goal is detected, the maximum reward of +2500 points is granted, as the attacker's main objective was achieved. An additional bonus is given if the ball is closer to the center, discouraging the agent from risking hitting the goalposts.

IV. Results and Discussion

The results of the testing scenarios were collected over many iterations to achieve more stable outcomes. However, the training process was lengthy, taking approximately 4 hours to complete 3 million steps (245 iterations), which limited our exploration possibilities. In the scenario without a goalie, the episode lengths and rewards over time during training are shown in Figures 11 and 12, respectively. Analysis of the graphs reveals that, despite the extended training times, the results were still not entirely stable, suggesting that with additional training time, the agent's performance could be further improved.

The distributed testing outcomes of the scenario with no goalie are presented in Figure 8. In the majority of situations, the attacker demonstrates remarkable effectiveness, with a success or near-success rate of approximately 75%.

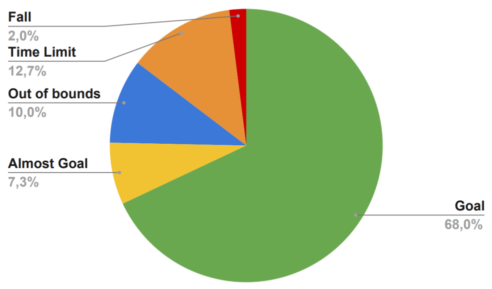

Fig. 8. Attacker Testing Results (without Goalie)

Only a small fraction of attempts (less than 15%) resulted in highly punishing outcomes, such as falls or time limit violations, suggesting a capable attacker agent. Interestingly, in 17% of the episodes, the robot displayed the "right intention," meaning that it chose an appropriate kicking attempt. Despite this, either the kicking position or the direction was misaligned, preventing a successful outcome.

A. Attacker vs Goalie without RL Adaptation

Additionally, we conducted experiments placing the attacker with the goalie without adapting the attacker's strategy to account for the goalkeeper's presence. At first, the goalie was too efficient because the shots were mostly perpendicular to the goal and so virtually no goals were scored. As such, we removed the goalie's option to not dive to defend a kick and got the results displayed in Figure 9.

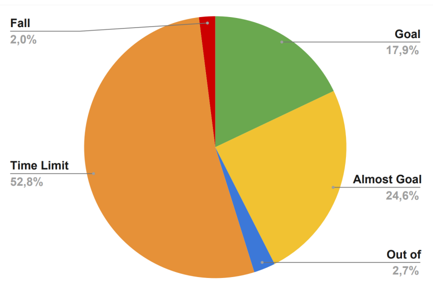

Fig. 9. Attacker Testing Results (without Goalie RL Adaptation)

The impact of adding a goalie, which the attacker was not prepared for, is very noticeable, as the goal rate dropped by 75%, being largely replaced by near misses, which was an expected outcome. Because the episodes were not terminated after a defense to avoid false positives, this naturally resulted in a significant increase in both average episode length and the number of episodes that ended due to the time limit, as shown in the comparative Figure 13.

B. Attacker vs Goalie with RL Adaptation

Finally, to train the attacker while accounting for the goalie's influence, several modifications were implemented. The observation space was expanded to include the (X, Y) coordinates of the goalie, increasing the total observation dimensions to 10 values. Adjustments were also made to the reward structure to guide the attacker more effectively. Penalties for not kicking the ball were aggravated to minimize aimless wandering, while the punishment for going out of bounds was alleviated. Additionally, the penalty for ending the episode due to time constraints was attenuated if the ball remained within the width of the goalposts, regardless of its depth distance to it. However, most notably of all, a major bonus reward was introduced for scoring a goal, with the reward increasing based on the distance of the ball from the goalie when it crossed the goal line. These changes aimed to incentivize goal-oriented behavior that tried to dodge the goalie's defense as much as possible. The testing results for this training are presented in Figure 10.

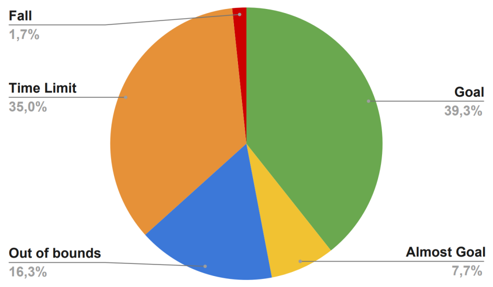

Fig. 10. Attacker Testing Results with Goalie RL Adaptation

Comparing the values with the previous scenarios, as shown in Figure 14, reveals that the training was quite effective, allowing the attacker to recover a significant portion of its earlier demonstrated potential. Nonetheless, further improvements are needed, as less than 50% of the strikes result in goals or near misses.

Training with two players on the field was significantly slower to emulate, which increased the overall training time. This constraint forced the team to complete only 123 iterations over more than three hours. As shown in Figures 15 and 16, the training process was unstable, requiring much longer sessions to refine and perfect the model's behavior.

C. Limitations and Shortcomings

During the development of the project, the team faced several constraints that impacted both our progress and the outcomes. A significant limitation was the lack of powerful computational resources, which restricted the efficiency and scale of training processes and the training configuration experimentation we could achieve during the narrow development time available. Moreover, the complexity of the forked codebase posed challenges, leading to lost time in understanding and modifying the system. Then, the finicky adjustment of observation and action spaces, as well as reward structures for the agent, added to the development complexity. We verified that the slight configuration changes yielded drastically agent results and that part of this variability in results stemmed from the starting training state, which introduced inconsistencies, further complicating the evaluation and refinement of the models.

V. Conclusions and Future Work

In this work, we examined how to train a humanoid robot to execute penalty kicks in the RoboCup 3D Simulation League using Deep Reinforcement Learning (DRL). We created an attacker agent that uses the Proximal Policy Optimization (PPO) algorithm to learn how to approach the ball, strategically position itself, and attempt to score goals. To assess the attacker's learning behavior, a heuristic-based goalie was used to mimic a realistic defending scenario. In no-goalie scenarios, the results are encouraging; nevertheless, the scoring rate declines when the defensive agent is present. After further tweaking the model and performing a new training, the attacker was indeed capable of adapting to the goalie and get good penalty goals. However, the results indicate that the model training could have been prolonged for more consistent and satisfying results.

Future work could focus on enhancing the attacker's adaptability through more sophisticated reward functions and additional training against more complex goalie strategies. Exploring simultaneous training of the attacker and goalie could also lead to favorable results.

References

[1] I. Noda, S. Suzuki, H. Matsubara, M. Asada, and H. Kitano, "Robocup-97: The first robot world cup soccer games and conferences," *AI magazine*, vol. 19, no. 3, pp. 49–49, 1998.

[2] Aldebaran Robotics, "Nao the humanoid and programmable robot," http://www.aldebaran.com/en/nao, last accessed 23 Dec 2024.

[3] M. Abreu, L. P. Reis, and N. Lau, "Designing a skilled soccer team for robocup: Exploring skill-set-primitives through reinforcement learning," *arXiv preprint arXiv:2312.14360*, 2023.

VI. Appendix

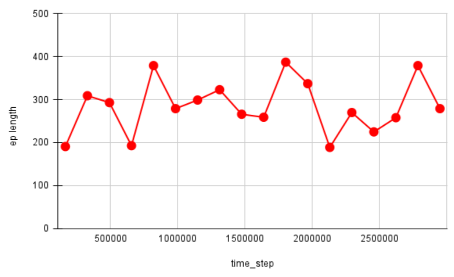

Fig. 11. Attacker RL Training Episode Length (without Goalie)

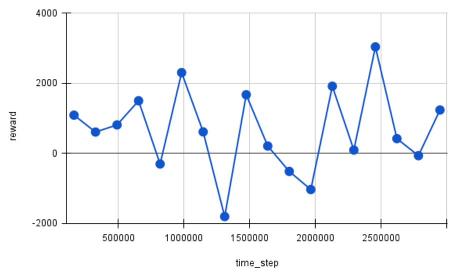

Fig. 12. Attacker RL Training Reward Results (without Goalie)

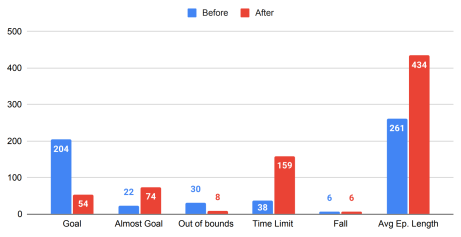

Fig. 13. Testing Results Comparison Between Before Goalie and After Introducing the Goalie without further RL Attacker Adaptation

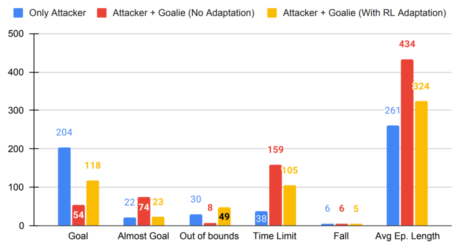

Fig. 14. Testing Results Comparison Between All 3 Different Experimented Scenarios

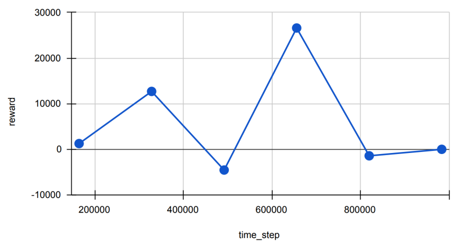

Fig. 15. Attacker RL with Goalie Training Reward Results

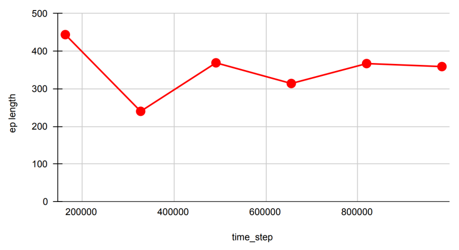

Fig. 16. Attacker RL with Goalie Training Episode Length