ITAndroids 2D Soccer Simulation Team Description Paper 2024

Davi M. Vasconcelos, Luiz F. B. Ramos, Marcos R. O. A. Maximo, Nean Segura, and Vinícius F. Almeida

Aeronautics Institute of Technology,
São José dos Campos, São Paulo, Brazil
{davi.muniz41,ramosbrito2004,seguranean,vinifreitas.d.a}@gmail.com,mmaximo@ita.br
itandroids-soccer2d@googlegroups.com
http://www.itandroids.com.br/

Abstract The ITAndroids 2D Soccer Simulation team is composed of undergraduate students of the Aeronautics Institute of Technology. This paper explores three topics: defensive behavior for 1-versus-1 situations with Deep Reinforcement Learning, improvement of the goalkeeper's positioning, and influence of the librcsc's version on the agent2d performance. The developed defensive behavior outperformed agent2d in the presented task. The novel goalkeeper's implementation dramatically decreased the loss rate of ITAndroids against RoboCIn. Furthermore, the agent2d performance considerably depended on the librcsc's version: the latest agent2d running with the most updated librcsc release achieved a win rate of 71.6% against agent2d 3.1.0 with librcsc 4.1.0.

1 Introduction

ITAndroids is a competitive robotics team from Aeronautics Institute of Technology reestablished in 2011. The group participates in the following leagues: RoboCup Soccer Simulation 2D (RCSS2D), RoboCup Soccer Simulation 3D, RoboCup Humanoid Kid-Size, IEEE Humanoid Robot Racing, IEEE Very Small Size, and RoboCup Small Size League.

Our RCSS2D team, ITAndroids 2D, has continuously participated in the Latin American Robotics Competition (LARC) and Brazilian Robotics Competition (CBR – acronym for *Competição Brasileira de Robótica*) since 2011. ITAndroids 2D competed in RoboCup from 2012 to 2023, except in 2014 and 2020. Table 1 shows the placements in these competitions.

2 Previous Works

Our code base uses agent2d [1] as the base team. Since 2011, we focused on improving mechanisms already present in the agent2d framework. We improved the action chain evaluator with Particle Swarm Optimization (PSO) [2]. Furthermore, we developed heuristics [3] to increase attack and defense performance:

Table 1: Placements of ITAndroids 2D in the previous RoboCup and LARC competitions.

Year	RoboCup	LARC
2023	10th	3rd
2022	11th	3rd
2021	11th	5th
2020	—	4th
2019	13th	1st
2018	9th	2nd
2017	15th	3rd
2016	13th	2nd
2015	13th	1st
2014	—	1st
2013	13th	1st
2012	10th	1st

type of formation (attack or defense) selection based on probability of scoring a goal, field evaluator selection based on opponent team, and defender optimal marking in opponent attack situations. Many early improvement ideas were inspired by HELIOS [4] and Nemesis [5].

The team proposed in 2018 a novel technique to determine the in-game ball possession [6]. We used a Finite-State Machine called Possession Automaton that takes into account the current and the last game situations to infer the ball possession. Since we do not estimate ball possession based on a single game cycle, we obtained a classification accuracy 18% higher than the default estimator of the base team.

We experimented with Deep Reinforcement Learning (DRL) techniques to improve the goalkeeper defense in penalty situations [7]. After five training experiments using the Proximal Policy Optimization (PPO) algorithm, we achieved a penalty defense rate of 40% against agent2d, twenty percent higher than the base team rate.

Our current efforts are developing individual defensive behaviors with DRL and improving the goalkeeper due to changes in the RoboCup Soccer Simulator Server (RCSSServer). We also investigated the influence of the librcsc's version on the performance of agent2d.

Several authors explored the development of individual behaviors with Reinforcement Learning (RL) and DRL in the RCSS2D domain. Gabel and Riedmiller [8] presented an RL ball interception behavior with approximately the same performance as a hand-made behavior. Gabel, Riedmiller, and Trost [9] developed a defensive behavior for 1-versus-1 situations with RL techniques. Carvalho and Oliveira [10] achieved a successful rate of 58% in the dribbling task against a single opponent. Zare *et al.* [11] demonstrated the capability of DRL algorithms to learn a 2-versus-1 defense task, where a single learning agent cooperates with its goalkeeper to protect the goal from the opponent with the ball.

3 Defensive Behavior with Deep Reinforcement Learning

Davi Vasconcelos [12], member of ITAndroids 2D, extended the work of Thomas Gabel, Martin Riedmiller, and Florian Trost [9] by learning the defense task with state-of-the-art DRL algorithms. In this section, we summarize the learning experiment and outline the main results of this work.

The defense task consists of two players, the learning agent and the opponent, where the learning player must get the ball from the opponent by tackling or kicking it. Table 2 describes the possible outcomes for the episode, where p_0 is the minimum catch probability for success, k_m is the kick margin, p_r is the player's radius, b_r is the ball's radius, k_t is a tolerance added for short opponent's kicks, d_0 is the minimum distance between learning agent and opponent for failure, and C_{max} is the number of cycles for a timeout. The success outcome arises when the player can issue the kick or tackle commands, the latter with a minimum probability of 0.8. The wrong outcome occurs when the opponent panics and kicks the ball away. The failure outcome happens when the opponent is too far from the learning player or the episode reaches the time limit and the opponent has the ball possession.

Table 2: Summary of the possible outcomes with their respective conditions for the presented task.

Outcome	Condition	Parameters
Success	$p_{\text{tackle}} \geq p_0$ or $d(\text{ball}, \text{player}) < k_m + p_r + b_r$	$p_0 = 0.8$ $k_m = 0.7$ $p_r = 0.3$ $b_r = 0.085$ $k_t = 1.0$ $d_0 = 7$ $C_{max} = 35$
Wrong	ball is out of field or $d(\text{ball}, \text{opponent}) > d_w$ where $d_w = k_m + k_t + p_r + b_r$	
Failure	$d(\text{player}, \text{opponent}) > d_0$ or (cycle $= C_{max}$ and $d(\text{ball}, \text{opponent}) \leq d_w$)	
Timeout	cycle $= C_{max}$	

In contrast to [9], we limited the training region to the midfield. The learning agent started in a random position inside a square of side 1.2 and center $(0, 0)$ with random velocity and body angle. The opponent initialization and observation space are as in [9]. We chose an action space of 144 elements. We obtained 36 turn actions by discretizing the interval $(-180°, 180°]$ using a $10°$ step. Likewise, we got 108 dash actions by discretizing the direction interval $(-180°, 180°]$ with a step of $10°$ for each power in $\{30, 60, 100\}$. Furthermore, we applied the reward function

$$r(s,a) = \begin{cases} 10 & \text{if } s \in S_s \\ -10 & \text{if } s \in S_f \\ -0.01 & \text{if } d(\boldsymbol{p}, \boldsymbol{m}) < 2 \\ -0.25 & \text{otherwise.} \end{cases}, \tag{1}$$

where S_s is the set of successful states, S_f is the set of failure states, $\boldsymbol{p}$ is the player's position, $\boldsymbol{m}$ is the mean between the opponent's position and ball's position, and $d(\boldsymbol{p}, \boldsymbol{m})$ is the distance between $\boldsymbol{p}$ and $\boldsymbol{m}$. Reward engineering was needed since the agent did not learn as expected without it after several runs.

We developed a custom tool in `C++` and Python to execute the training. The learning agent is librcsc-based and exchanges messages with the Python program by sending observations and requesting actions through Protobuf over a TCP socket. The Python program implements an OpenAI Gym [13] and uses the RLlib package [14] for the training algorithms. We selected the Rainbow algorithm [15] for training and excluded part of its features due to computational time limitations. We kept the following Rainbow's improvements over the Deep Q-Networks (DQN): Prioritized Experience Replay, Double DQN, and Dueling DQN. For the action-value function approximation, we used four layers of 256 neurons for the advantage and value networks and one layer of 256 neurons for the combiner layer.

We ran three independent runs of 5M time steps for the opponents Thunder-League and YuShan in full-state mode. Figure 1 depicts the learning curves. The mean reward converged to a positive value with reasonably low variance, which suggests stable learning. Figure 2 shows the cumulative average of the outcome rates for the runs in Figure 1. The number of episodes of each run differed as we kept the maximum time step at 5M. Thus, we adjusted the time series before plotting by cutting them at the size of the minimum time series between the runs.

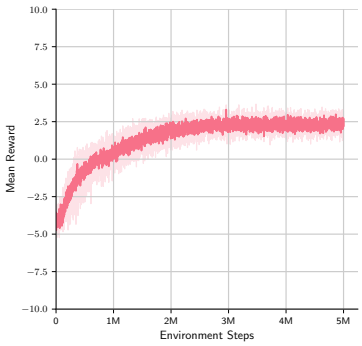
As shown in Figure 2, the success curves increased while the failure curves decreased over time. The wrong curves initially increased but gradually decreased after several episodes. At first, the agent does not know how to get closer to the opponent. After it learns how to move toward the opponent, the opponent's panic cases increase. Later, the agent learns to get closer to the opponent without leading to the opponent's panic, which reduces wrong outcomes.

Table 3 presents the performance of the trained models and agent2d against ThunderLeague and Yushan in the defense task. π_{tl} and π_{ys} represent the policy obtained after training against ThunderLeague and YuShan, respectively. We randomly selected the policies from the respective runs previously presented. The learned policies outperformed agent2d by a large margin. A satisfactory generalization was obtained as the success rates of the policies were nearly equal. Both opponents are agent2d-based teams, which explains the generalization to a certain degree. We tried to evaluate the models against teams not based on agent2d (FRA-UNIted and Oxsy), but we could not run their binaries.

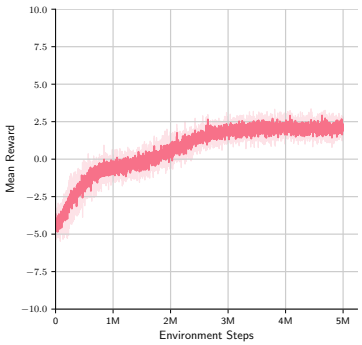
A demonstration video is available at `https://www.youtube.com/watch?v=slMQXwzra5Y`. The video shows the policy before and after training against YuShan.

4 Goalkeeper Fixes and Improvements

The RCSSServer underwent significant changes in version 17. The dash and catch models were modified. An agent can no longer issue a negative dash power to

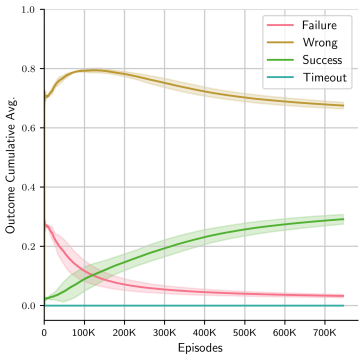


(a) Soccer player learns against ThunderLeague.

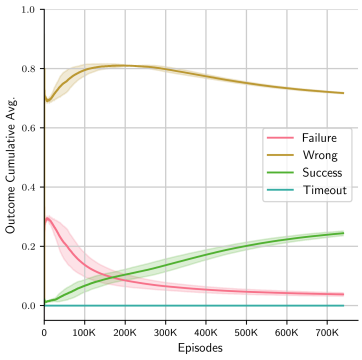


(b) Soccer player learns against YuShan.

Figure 1: Learning curve against ThunderLeague and YuShan. The shaded region represents the 95% confidence interval for the episode's mean reward in three runs of 5M environment steps. We evaluated the moving mean using 1000 environment steps.



(a) Soccer player learns against ThunderLeague.



(b) Soccer player learns against YuShan.

Figure 2: Cumulative average of the outcome rates against different opponents. The shaded region represents two times the standard error around the mean for the runs of Figure 1.

Table 3: Outcome rates after running the trained models and agent2d against two opponents in full-state mode. We estimated the rates in 50K episodes.

Opponent	Player	Success	Failure	Wrong
ThunderLeague	π_{tl}	35.47%	1.69%	62.84%
	π_{ys}	35.52%	2.30%	62.18%
	agent2d	13.98%	25.80%	60.22%
YuShan	π_{tl}	32.25%	2.00%	65.75%
	π_{ys}	33.23%	2.16%	64.61%
	agent2d	19.65%	51.15%	29.20%

accelerate backward. Consequently, this modification broke most `doDash` calls, especially for the goalkeeper, when agents attempted to execute a back dash.

In the latest simulator's version, the omnidirectional dash model should be used to accelerate backward by selecting a dash angle of 180°. Hence, we changed the `doDash` calls to the omnidirectional model to fix the back dashes.

The goalkeeper of agent2d tries to maintain the same y-coordinate as the ball — restricted to the goalposts. Thus, the goalkeeper leaves a significant space free, notably when the ball is close to the goal.

We present a novel goalkeeper heuristic by considering how a goalkeeper should behave in real soccer matches. When our team attacks on the opponent's side, the goalkeeper should be advanced, preventing the opponent midfield from completing a line-breaking pass between our defensive line, creating a goal-scoring opportunity. When our team is defending, the goalie should be positioned to cover the goal with the widest angle possible — which was not implemented in agent2d.

With these requirements in mind, we modeled the goalkeeper's position according to the Bisector Intersection Positioning (BIP). The BIP operates as follows: first, we calculate the x-coordinate of a vertical line as the weighted average of the ball's x-coordinate and pitch half-length. Next, we evaluate the bisector of the angle determined by the left goal post, ball, and right goal post. Finally, we set the goalkeeper's position to the intersection between the bisector and vertical line. Furthermore, we manually adjusted extreme positions, such as the ball being inside the goal box, to minimize the angle for attacking opponents as much as possible.

Table 4 shows the performance of ITAndroids against RoboCIn after and before the cited changes in 500 matches. We also modified the team formation, improving the defense formation. The loss rate dramatically decreased.

Table 4: Performance of ITAndroids against RoboCIn before and after the improvements in 500 matches.

Version	Win rate	Loss rate	Draw rate
Before changes	4.8%	86.6%	8.6%
After changes	11.4%	49.2%	39.4%

5 Impact of librcsc's Version on Team Performance

The agent2d base team has considerable functionalities implemented in the librcsc [1]. The librcsc provides a collection of classes and methods useful for any soccer player of the RoboCup 2D Soccer Simulation League. It also contains mathematical models of the RCSSServer: the success probability of a tackle command issued by an agent, for instance. Hence, the usage of an outdated librcsc may reduce the team performance due to changes in the RCSSServer.

Our team uses the librcsc 4.1.0 along with agent2d 3.1.0. Since the comeback of ITAndroids 2D in 2011, we have not updated both software. Furthermore, we decided to mix the source codes of librcsc and agent2d in a single repository to facilitate improvements in the librcsc. However, previous members modified librcsc's files without properly specifying the changes on them. Therefore, updating our librcsc is not a trivial task because we will need to track those modifications and apply them in novel files. Before doing such laborious work, we investigated the relevance of different versions of librcsc to the performance of agent2d.

We compared the source codes of two versions of agent2d: 3.1.0 and the latest release on GitHub, which is called support-v18. Both code bases are nearly identical. We did not identify significant changes but minor refactoring. Hence, any noticeable performance difference between the teams could be explained by changes in the librcsc. The support-v18 version used the librcsc release of the same name, whereas the 3.1.0 version used the librcsc 4.1.0.

We ran 500 matches between the teams without extra time and penalties in RCSSServer 18.1.3. Table 5 contains the match results. The most updated agent2d release outperformed agent2d 3.1.0 with a win rate of 71.6%. Thus, the librcsc's version had a tremendous impact on team performance.

Table 5: Performance of agent2d support-v18 against agent2d 3.1.0 in 500 matches.

Win rate	Loss rate	Draw rate
71.6%	12.0%	16.4%

6 Conclusions and Future Work

This paper presented the recent efforts of ITAndroids 2D. We developed a defensive behavior using DRL for 1-versus-1 situations. Also, we switched the dash calls to the omnidirectional model and reworked the default implementation of the goalkeeper positioning of the agent2d. Furthermore, we measured the impact of the librcsc's version on the agent2d team performance. In the future, we aim to run the learning experiment with other DRL algorithms, update our librcsc, and deploy the learned defensive behavior.

7 Acknowledgements

We would like to acknowledge the RoboCup 2D Soccer Simulation community for sharing their developments. In particular, we would like to acknowledge Hidehisa Akiyama for agent2d, librcsc, soccerwindow2, and fedit2 [1]. Finally, we acknowledge our sponsors: Altium, CENIC, Field Pro, Intel, ITAEx, MathWorks, Micropress, Neofield, Polimold, Rapid, SIATT, SolidWorks, STMicroelectronics, Virtual Pyxis, and Visiona Space Technology.

References

[1] Hidehisa Akiyama and Tomoharu Nakashima. "HELIOS Base: An Open Source Package for the RoboCup Soccer 2D Simulation". In: *The 17th annual RoboCup International Symposium*. July 2013.

[2] Fábio Mello et al. "ITAndroids 2D Soccer Simulation Team Description 2012" (2012).

[3] Felipe Coimbra et al. "ITAndroids 2D Soccer Simulation Team Description Paper 2017" (2017).

[4] Hidehisa Akiyama and Hiroki Shimora. "HELIOS2010 Team Description" (2010).

[5] Mehrab Norouzitallab et al. "Nemesis Team Description 2010" (2010).

[6] Felipe Coimbra and Lucas Lema. "ITAndroids 2D Soccer Simulation Team Description 2018" (2018).

[7] Diego Fidalgo et al. "ITAndroids 2D Soccer Simulation Team Description Paper 2020" (2020).

[8] Thomas Gabel and Martin Riedmiller. "Learning a Partial Behavior for a Competitive Robotic Soccer Agent." *KI* 20 (Jan. 2006), pp. 18–23.

[9] Thomas Gabel, Martin Riedmiller, and Florian Trost. "A Case Study on Improving Defense Behavior in Soccer Simulation 2D: The NeuroHassle Approach". In: *RoboCup 2008: Robot Soccer World Cup XII*. Ed. by Luca Iocchi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 61–72. ISBN: 978-3-642-02921-9.

[10] Arthur Carvalho and Renato Oliveira. "Reinforcement learning for the soccer dribbling task". In: *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, Aug. 2011. DOI: 10.1109/cig.2011.6031994. URL: http://dx.doi.org/10.1109/CIG.2011.6031994.

[11] Nader Zare et al. "CYRUS 2D Simulation 2019. Team Description Paper" (2019).

[12] Davi Vasconcelos. "Learning Individual Behaviors for Simulated Robot Soccer". B.Sc. Thesis. São José dos Campos: Aeronautics Institute of Technology, 2023.

[13] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].

[14] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. 2018. arXiv: 1712.09381 [cs.AI].

[15] Matteo Hessel et al. *Rainbow: Combining Improvements in Deep Reinforcement Learning*. 2017. arXiv: 1710.02298 [cs.AI].