FC Portugal 3D Simulation Team: Team Description Paper 2023

Miguel Abreu[1][0000−0002−6342−2054], Luís Paulo Reis[1][0000−0002−4709−1718], and Nuno Lau[2][0000−0003−0513−158X]

[1] LIACC/LASI/FEUP, Artificial Intelligence and Computer Science Lab, Faculty of Engineering of the University of Porto, University of Porto, Portugal
{m.abreu,lpreis}@fe.up.pt

[2] IEETA/LASI/DETI University of Aveiro 3810-193 Aveiro, Portugal
nunolau@ua.pt

Abstract. FC Portugal, a team from the universities of Porto and Aveiro, in Portugal, won the main competition of the 2022 RoboCup 3D Simulation League, with 17 wins, 1 tie and no losses. FC Portugal also won the 2022 RoboCup 3D Simulation League Technical Challenge, accumulating the maximum amount of points by ending first in its both events: the Free/Scientific Challenge, and the Fat Proxy Challenge. The team presented in this year's competition was rebuilt from the ground up since the last RoboCup. No previous code was used or adapted, with the exception of the 6D pose estimation algorithm, and the get-up behaviors, which were re-optimized. This paper describes the team's new architecture and development approach. Key strategy elements include team coordination, role management, formation, communication, skill management and path planning. New lower-level skills were based on a deterministic analytic model and a shallow neural network that learned residual dynamics through reinforcement learning. This process, together with an overlapped learning approach, improved seamless transitions, learning time, and the behavior in terms of efficiency and stability. In comparison with the previous team, the omnidirectional walk is stabler and went from 0.70 m/s to 0.90 m/s, the long kick from 15 m to 19 m, and the new close-control dribble reaches up to 1.41 m/s. The preparation for 2023 includes adapting to the new 3D simulation rules and use the new python robust source code to continue to build a reinforcement learning based team and foster high-level multi-agent coordination strategies, goalkeeper skills, general ball handling skills, and more.

1 Introduction

Historically, FC Portugal has contributed to the simulation league (2D and 3D) in numerous ways, including competitive methodologies and server improvements. [1] In the last years, the team has been focused on developing low-level skills and methodologies that leverage model knowledge to design more efficient

[1] For previous contributions concerning coaching, visual debugging, team coordination, sim-to-real, optimization algorithms and frameworks please refer to https://tdp.robocup.org/tdp/2022-tdp-fcportugal3d-robocupsoccer-simulation-3d/

reinforcement learning (RL) techniques [1–3, 6–9, 12, 13]. In 2019, it introduced the first running behavior in the league, which was learned from scratch using RL. After that, the development focus was on leveraging the robot's symmetry and analytical models to improve the learning efficiency, producing human-like skills in less time. However, due to the intensive use of high-level optimization algorithms, the low-level C++ code of the team grew more convoluted with a network of interrelated skills and the addition of environment frameworks to bridge the gap between C++ and Python. Maintenance complexity along with lack of compatibility of some libraries with modern Linux distributions led to a turning point in 2021.

After RoboCup 2021, we decided to rebuild all the code from the ground up in Python, without using or adapting previous code, with the exception of the 6D pose estimation algorithm [4], and the get-up behaviors, which were re-optimized. The new code is compatible with most data science libraries and machine learning repositories, allowing for fast development of new behaviors and tactics. Due to hardware improvements in recent years, developing a team in Python is no longer a major concern in terms of computational efficiency. However, some computationally demanding modules were written in C++ to ensure the agent is always synchronized with the server.

2 3D Simulation League

The RoboCup 3D simulation league uses SimSpark [14] as its physical multiagent simulator, which is based on the Open Dynamics Engine library. The league's environment is a 30 m by 20 m soccer field containing several landmarks that can be used for self-localization: goalposts, corner flags and lines. Each team consists of 11 humanoid robots modeled after the NAO robot. Agents get internal data (joints, accelerometer, gyroscope and foot pressure sensors) with a 1-step delay every 0.02 s and visual data (restricted to a 120° vision cone) every 0.06 s. Agents can send messages to teammates every 0.04 s (see Section 3.3 for further details). There are 5 humanoid robot types with 22 to 24 controllable joints, and slightly different physical characteristics. Each team must use at least 3 different types during an official game. [2]

3 Team Description

An overview of the agent and server can be seen in Fig. 1. The white and green modules are implemented in Python and C++, respectively. The agent takes advantage of Python's development speed and compatibility with major data science libraries, and C++'s performance for time-sensitive modules.

The server is responsible for updating the soccer environment, based on the actions received from all agents. It adds noise to the world state in the form of a calibration error that is constant and affects the robot's vision sensor, a

[2] The official rules can be found at https://ssim.robocup.org/3d-simulation/3d-rules/

perception error that follows a normal distribution and affects the coordinates of visible objects according to their distance, and a rounding error introduced by a lossy transmitter, where all numbers (including coordinates, joint angles and sensor readings) are rounded to the nearest hundredth.

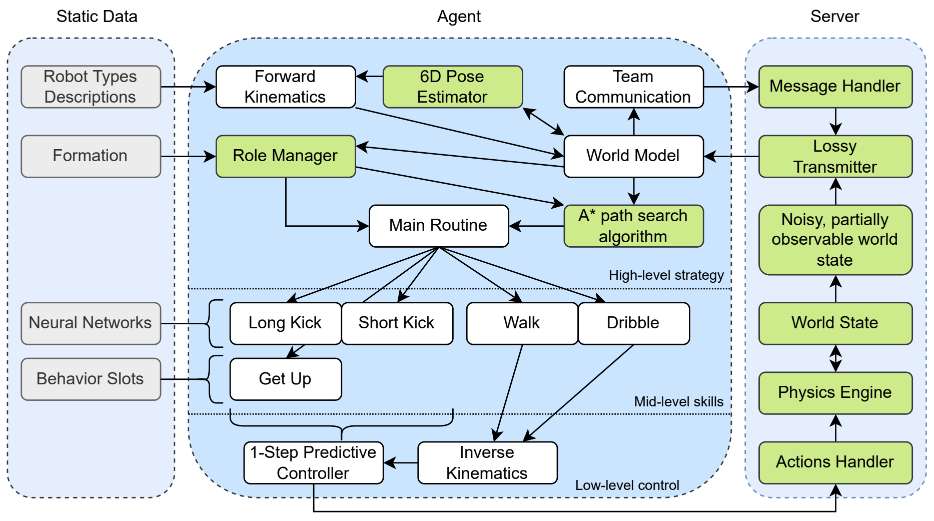

Fig. 1. Overview of the internal decision flow of agent and server, and their external interaction. White modules are implemented in Python while green modules are implemented in C++. Gray blocks represent static data.

As aforementioned, the agent receives internal data from the server every 20 ms, and visual data every 60 ms. The latter is a noisy partial view of the world state, which is fed to a 6D pose estimator to extract the robot's localization and orientation in a three-dimensional space. The algorithm leverages the known noise models to maximize the probability of the current map of perceived objects, by iteratively adjusting the robot's 6D pose. For an extensive description of this process please refer to Abreu et al. [4]. Forward kinematics is then used to estimate the pose of every body part for a given robot type. This self-awareness ability in conjunction with team communication allows each agent to have a reliable representation of the world state. The following paragraphs will describe the main components of the agent.

3.1 Low-level control

The low-level control consists of an inverse kinematics module, used by the Walk and Dribble, and a 1-step predictive controller, which is indirectly used by the same skills, and directly used by the Kick and Get Up. The inverse kinematics module simplifies locomotion by abstracting the joint information from the skill. The Kick and Get Up do not use this module, as it did not improve the final behavior performance. Since the server sends the observations with a 1-step delay, the current robot state can only be estimated by combining the previous observations with the last sent actions. This technique is nearly optimal when the actuators work below their maximum torque specifications, although the lossy transmission protocol impedes actual optimality.

3.2 Mid-level skills

As depicted in Fig. 1, the team has four major skills:

- **Kick**: A short kick is mainly used for passes, with a controllable range between 3 and 9 m. The long kick is generally used for shooting, and has an average distance of 17 to 19 m, depending on the robot type;
- **Walk**: The omnidirectional walk is able to sustain an average linear speed between 0.70 and 0.90 m/s, depending on robot type and walking direction;
- **Dribble**: The dribble skill pushes the ball forward, retaining close control with an avg. max. speed of 1.25 to 1.41 m/s, depending on robot type. The walk skill can push the ball but it is slower and provides no close control;
- **Get Up**: The robot uses this skill to get up after falling to the ground. There are three variations of the skill per robot type depending on the falling direction (front, back, side).

The model architecture used for all skills that rely on neural networks can be seen in Fig. 2. An underlying base model is used to guide the optimization at an early stage, expediting the learning process and improving the quality and human-likeness of the final behavior. For the Walk and Dribble skills, the underlying model is based on a Linear Inverted Pendulum (LIP) solution, which generates a cyclic walk-in-place behavior. The kick skill is built upon a hand-tuned base model which is divided into two sections: back swing and forward acceleration. The state of the base model is fed to the neural network as a single integer variable. The state of the robot comprises its internal sensors, position and velocity of joints, head height and a step counter. Depending on the skill, the target denotes a set of variables that encode small variations of the main objective, such as direction and distance. The relative position of the ball is always required except for the Walk skill.

The output of the shallow neural network (single hidden layer with 64 neurons) is added to the output of the base model to generate a target position for each feet and hand. As shown in Fig. 1, only the Walk and Dribble skills generate relative positions, thus requiring the inverse kinematics module to obtain target joint angles. The optimization is performed by the Proximal Policy Optimization algorithm [11], extended with Proximal Symmetry Loss [7].

3.3 Team communication

According to official rules, team communication is restricted to messages of 20 characters, taken from the ASCII subset [0x21, 0x7E], except for normal brackets

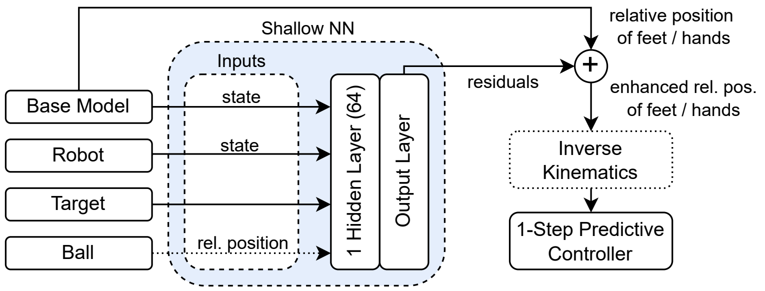

Fig. 2. Model architecture for the Kick, Walk, and Dribble skills

{0x28,0x29}. However, due to server limitations, some characters create undefined behaviors, such as single (0x27) or double (0x22) quotation marks, backslash (0x5C), and semicolon (0x3B) at the beginning of messages. Therefore, FC Portugal uses 88 alternatives for the first character and 89 alternatives for the remaining 19 characters, since the semicolon is included. This gives about $9.6{\times}10^{38}$ combinations per message.

The team uses three consecutive messages to send all the desired information. Accordingly, there are three distinct groups of variables, as seen in Table 1. Group A contains the position of the ball, and the position and state of teammates number 10 and 11, and opponents from number 7 to 11. The state is a Boolean indicating whether the agent has fallen. The position of a teammate has a precision of 10 cm and $x \in [-16, 16]$, $y \in [-11, 11]$, yielding $321 * 221 = 70941$ combinations. Considering the position and state of the robot, each teammate represents 141882 combinations. The opponents' position has the same range but lower precision (16 cm in x and 20 cm in y), generating $201 * 111 * 2 = 44622$ (including the state). Finally, the position of the ball has a precision of 10 cm and $x \in [-15, 15]$, $y \in [-10, 10]$, resulting in $301 * 201 = 60501$ combinations. Group B contains information about 7 teammates, and group C combines 2 teammates and 7 opponents.

Table 1. Description of message groups for the team communication protocol

Group	Teammates	Opponents	Ball	Combinations
A	10,11	7-11	Yes	141882ˆ2*44622ˆ5*60501=2.2e38
B	1-7	None	No	141882ˆ7=1.2e36
C	8,9	1-6	No	141882ˆ2*44622ˆ6=1.6e38

The design of each group followed two principles: maximize the amount of information per message, and gather entities that are usually seen together on the field. Every agent can send messages at a given time step. However, the server only broadcasts one of the messages to the whole team. Therefore, agents should only try to communicate if they have relevant and up-to-date data. A naive rule would be to only send a message if all the elements of a given group were recently seen. However, as an example, if group A was always visible, it could monopolize the communication medium, or if teammate number 5 was far from all the others, group B would never be sent. To solve these issues, a protocol was proposed, as seen in Table 2.

Table 2. Team communication protocol

Time Step	Round	Group	Max. RLP	Max. MP	Ball
0.00 s	1	A	0	0	is visible
0.04 s		B			-
0.08 s		C			-
0.12 s	2	A	1	-	is visible
0.16 s		B			-
0.20 s		C			-
0.24 s	3	A	2	-	is visible
0.28 s		B			-
0.32 s		C			-
0.36 s	1	...	...	...	...

Message groups are synchronized with the game time, which is the provided to all players by the server. There are three rounds of messages that are repeated every 0.36 s. Each round contains messages from all groups in a sequential order: A,B,C. Round 1 is the strictest, since an agent must see all the elements of the current group before it is allowed to broadcast. This means that the maximum number of recently lost players (RLP) and missing players (MP) is zero. The former includes player not seen in the last 0.36 s and the latter encompasses players that have not been seen in the last 3 s. Rounds 2 and 3 allow any number of missing players, but apply restrictions to the number of recently lost players. During a soccer match, missing players are typically far from the field or have crashed and left the server. In both cases, it is generally safe to ignore them. The main purpose of having rounds with different restrictions is to force high quality updates in round 1, and, if that is currently not possible, allow some losses in rounds 2 and 3. If a group is fully visible by any agent, all teammates will receive new information every 0.12 s, even if there are missing players; if one of the elements was recently lost, all teammates will be informed twice every 0.36 s; and if no agent has recently lost less than 2 elements from the current group, all teammates will get a broadcast once every 0.36 s.

3.4 Role Manager

The role manager is responsible for dynamically assigning roles to each player. Fig. 3 shows an overview of this process. The formation, shown on the left, indicates the desired position of each role, not the current position of each teammate. The red circles denote the actual position of the opponents. This example shows the formation when the team is attacking, i.e. the ball is closer to our team. The displayed roles are: goalkeeper (GK), central back (CB), man-marking (MM), left and right support (LS, RS), left and right front (LF, RF). The support roles are always close to the active player (AP) to assist when ball possession is lost, while the front roles are always closer to the opponent's goal to receive long passes. The man-marking roles mark the closest opponents to our goal in a sticky way, to avoid frequent switching.

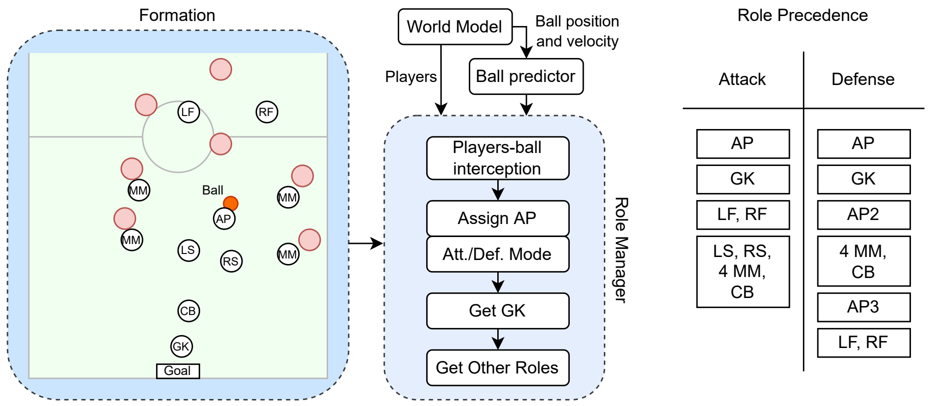

Fig. 3. Overview of the role manager. The formation (left) represents the desired position of each role, depending on the ball and opponents. The decision flow (middle) describes how the role manager assigns roles according to their precedence (right).

The decision flow of the role manager is shown on the middle of Fig. 3. The world model provides the position and velocity of all players and ball. The ball predictor yields a sequence of ball positions for future time steps, until the ball stops completely. This information is combined with the maximum walking speed to estimate when and where each player would intercept the ball. At this point, it is possible to answer which player should go to the ball, and whether the opponent will get there faster. If the latter is true, the defense mode is activated, changing the role precedence, as indicated on the right of Fig. 3.

The roles with highest priority are the AP followed by the GK. Therefore, if the closest player to the ball is the goalkeeper, it becomes the AP, and the GK role will be assigned to another player. If the team is attacking, the next roles are LF and RF, to create space and supporting angles for the AP. Then, the low priority roles are assigned: LS, RS, 4 MM and CB. This assignment is an optimization problem where the objective is to reduce the distance between the player and the assigned role. However, the man-marking roles have different weights, according to the distance of the marked opponent to our goal. In general, the optimization algorithm prioritizes dangerous opponents, and gives less importance to the CB, in relation to the LS and RS roles. If the team is defending, after the AP and GK comes the Active Player 2 (AP2), and later the Active Player 3 (AP3). The position of these roles is the ball position. Three player will try to intercept it until at least one of them is closer than the closest opponent. In defense mode, there are no support roles, and the front roles are the least important.

3.5 Path Planning

The path search algorithm is based on A* [5] and divides the soccer field into 70941 nodes (32 m $\times$ 22 m, 10 cm grid size). It is only employed when the player is not able to walk to the objective in a straight line. When dribbling the ball, the player is only allowed inside the field, reducing the work area to 30 m $\times$ 20 m. Static obstacles include the goalposts and the goal net, while dynamic obstacles include the players and the ball. Dynamic obstacles are identified by a 4-tuple $\langle$position, hard radius, soft radius, cost$\rangle$, where the hard radius defines the inaccessible region around the obstacle, and the soft radius defines a region with an additional radially decreasing cost for the path search algorithm. When defining a player as an obstacle, the hard radius takes into consideration the position of each arm and leg, when possible, to avoid colliding with fallen players. The soft radius depends on several factors:

- Is the agent walking or dribbling? If dribbling, the radius of other players increases, since the maneuverability decreases;
- The radius of an opponent decreases when closer to the ball, to allow tackling;
- The radius of a teammate increases if it was assigned a more important role.

Regarding the last factor, the role precedence introduced in Fig. 3 is not only used to assign roles to players. It is also important to ensure some additional space is given to higher priority roles. As an example, when the team is defending and there are 3 active players, AP2 will give extra space to AP (through a larger soft radius), and AP3 will get away from AP and AP2. This also applies to all the other roles. Players are not considered obstacles if their distance is over 4 m or they have not been seen (or perceived through team communication) for longer than 0.5 s.

In order to achieve path stability, the initial path position is obtained by estimating the position of the robot after 0.5 s, considering the current velocity as a constant. To extract the current target for walking or dribbling, the number of considered path segments depends on the current speed of the robot. This technique generates a smooth path without losing too much accuracy.

This module was implemented in C++ due to its computational complexity. To ensure that no simulation cycle is lost, the path planning algorithm runs until the end or until a timeout (5 ms) expires. In the latter case, the best current path is returned. Therefore, while running asynchronously with the server, the performance of the team depends on the computational power of the host machine, but not to the point where it misses a significant amount of cycles, except for abnormally slow host machines.

3.6 Main Routine

The main routine is divided into 5 major steps as depicted in Fig. 4. The agent starts by selecting a general intention from: move to some point, dribble the ball, push the ball while walking, kick to pass or shoot, get up, or beam before a kickoff. Some situations require a specific intention: the active player is required to kick the ball during kickoff and other play modes, fallen players must get up, inactive players must move to strategic position, etc. However, the active player faces more complex scenarios where it must decide whether to dribble, push or kick the ball.

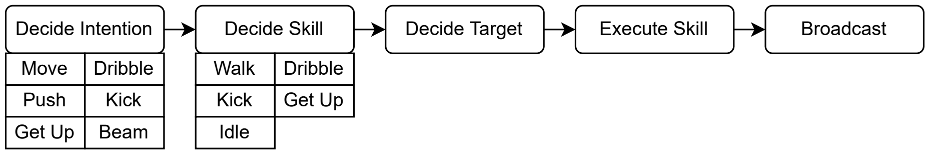

Fig. 4. Major steps of the agent's main routine

To reach a verdict, the algorithm scores multiple passes along a grid of feasible options, and shooting alternatives if the goal is reachable, considering collisions with non-fallen players, space to kick, orientation convenience for the kicker, and error in the kick direction, which is more relevant as the distance increases. If shooting is likely to be successful, the decision is immediate. Otherwise, passing is only preferred if it allows a faster game progression than dribbling. When there is no space to kick or dribble, the robot will push the ball by walking towards it.

The second step is to decide the current skill. As an example, if the agent decides to kick or dribble, it must first walk until it is in the correct position to use the desired skills. Then the agent must compute a target for the selected skill, if required, e.g., a target position and orientation while walking. Finally, the skill is executed, by applying the respective model and feeding the output to the low-level controller. An extra step broadcasts visual information to other teammates if the conditions are met, as explained in Section 3.3.

4 Competition Format

The main league competition was organized as shown in Fig. 5. The seeding round groups were drawn randomly, after the best two teams from the previous year were separated. Group A and B played a round-robin tournament that dictated which teams were drawn for group C and D in round 1. In round 1, 2 and 3, the last two teams were eliminated. The last two teams in round 1 and 2 played for 7^{th} to 10^{th} place in group H, and the last two teams in round 3 played again for the 5^{th} place. The four qualified teams in round 3 played the semi-finals. As expected, the winners faced each other in the final and the losers played for the 3^{rd} place.

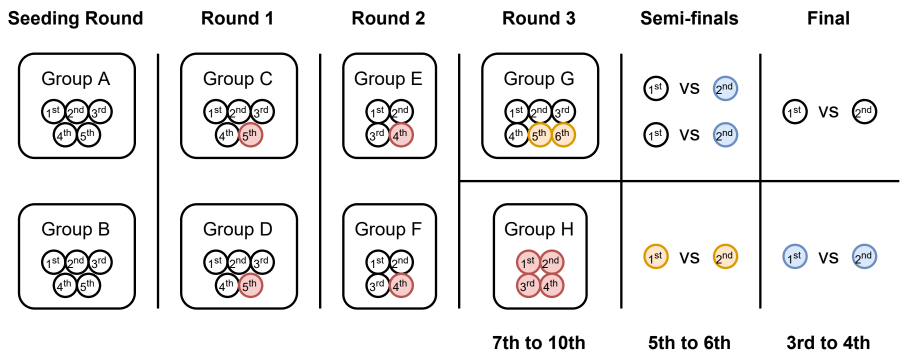

Fig. 5. Main competition format

In addition to the main competition, there was a Technical Challenge, which was composed of two events: the Free/Scientific Challenge, where competing teams presented their research work in the context of the 3D simulation league; and the Fat Proxy Challenge, which was played in a single round-robin tournament using the magmaFatProxy [10]. The purpose of the proxy is to provide the same skills to all teams, preventing the agent from controlling the joints of the robot, thus making the competition exclusively about high-level strategies. The winner of the Technical Challenge is the team that accumulates more points in both events.

5 Results

The results of the main competition are summarized in Fig. 6. FC Portugal finished in first place without any loss, having 17 wins and only 1 tie during the seeding round. [3] It managed to end in first place in the respective group of all round-robin rounds. During the course of the competition, the team scored 84 goals while conceding only 2. In the final, it defeated the strong magmaOffenburg team by 6-1.

Table 3 shows the scores obtained in the Free/Scientific and Fat Proxy challenges by each participating team. As aforementioned, the Technical Challenge is won by the team that accumulates more points in both events. In each case, the number of awarded points is $25 - 20 * (rank - 1)/(number\_of\_participants - 1)$.

FC Portugal won both the Free/Scientific Challenge and the Fat Proxy Challenge, leading to a victory in the overall Technical Challenge with 50 points. The presented scientific contribution introduced the dribble skill — the league's first close control dribble behavior, reaching speeds of up to 1.41 m/s. In the Fat Proxy Challenge, the team registered 4 wins, no ties and no losses, 21 scored goals and 3 conceded.

[3] Official results can be found at https://cloud.robocup.org/s/ifX7TDsaHpCFWWH

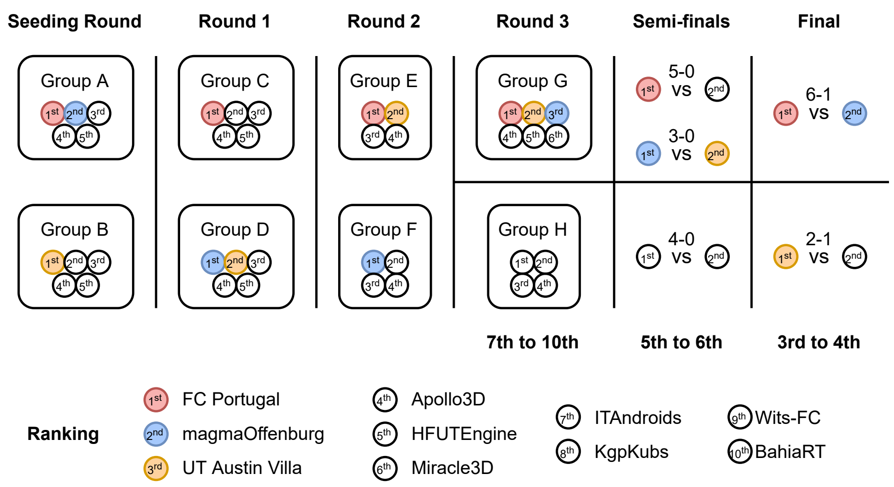

Fig. 6. Main competition results

Table 3. Technical Challenge results based on the number of points accumulated in the Free/Scientific Challenge and the Fat Proxy Challenge

Team	Overall		Free/Scientific		Fat Proxy	
	Rank	Points	Rank	Points	Rank	Points
FC Portugal	**1**	**50**	**1**	**25**	**1**	**25**
magmaOffenburg	2	25	3	5	2	20
BahiaRT	3	20	2	15	5	5
UT Austin Villa	4	15	–	–	3	15
Wits-FC	5	10	–	–	4	10

6 Conclusion

FC Portugal has developed numerous skills and methodologies concerning the NAO humanoid robot and the simulation league in general. Currently, the team has a very robust code base developed from scratch after RoboCup 2021, which led to a victory in the 2022 RoboCup 3D simulation league, as well as the Free/Scientific Challenge, the Fat Proxy Challenge, and consequently, the Technical Challenge. In the main competition it registered 17 wins, 1 tie, 0 losses, 84 goals scored, only 2 conceded, and in the Fat Proxy Challenge 4 wins, 0 ties, 0 losses, 21 goals scored, and 3 conceded.

An integrated learning approach guaranteed that skills such as the omnidirectional walk, dribble and kick can attain high performance but also smooth transitions, without falling or requiring intermediate steps. Despite the considerable gain in performance and competitiveness, in comparison with previous years, there are still many improvement opportunities. Future research directions include high-level multi-agent coordination strategies, opponent modeling, goalkeeper skills, omnidirectional kicks, optimization algorithms, and more.

References

1. Abdolmaleki, A., Simões, D., Lau, N., Reis, L.P., Neumann, G.: Learning a humanoid kick with controlled distance. In: RoboCup 2016: Robot World Cup XX. pp. 45–57. Springer (2016)
2. Abreu, M., Lau, N., Sousa, A., Reis, L.P.: Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). pp. 256–263. IEEE (2019)
3. Abreu, M., Reis, L.P., Lau, N.: Learning to run faster in a humanoid robot soccer environment through reinforcement learning. In: RoboCup 2019: Robot World Cup XXIII. pp. 3–15. Springer (2019)
4. Abreu, M., Silva, T., Teixeira, H., Reis, L.P., Lau, N.: 6d localization and kicking for humanoid robotic soccer. Journal of Intelligent & Robotic Systems **102**(2), 1–25 (2021)
5. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968). https://doi.org/10.1109/tssc.1968.300136, https://doi.org/10.1109/tssc.1968.300136
6. Kasaei, M., Abreu, M., Lau, N., Pereira, A., Reis, L.P.: Learning hybrid locomotion skills – learn to exploit residual dynamics and modulate model-based gait control. arXiv preprint arXiv:2011.13798 (2020)
7. Kasaei, M., Abreu, M., Lau, N., Pereira, A., Reis, L.P.: A cpg-based agile and versatile locomotion framework using proximal symmetry loss. arXiv preprint arXiv:2103.00928 (2021)
8. Kasaei, M., Abreu, M., Lau, N., Pereira, A., Reis, L.P.: Robust biped locomotion using deep reinforcement learning on top of an analytical control approach. Robotics and Autonomous Systems **146**, 103900 (2021)
9. Kasaei, S.M., Simões, D., Lau, N., Pereira, A.: A hybrid zmp-cpg based walk engine for biped robots. In: Iberian Robotics conference. pp. 743–755. Springer (2017)
10. magmaOffenburg: magmafatproxy. https://github.com/magmaOffenburg/magmaFatProxy (2022)
11. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
12. Simões, D., Amaro, P., Silva, T., Lau, N., Reis, L.P.: Learning low-level behaviors and high-level strategies in humanoid soccer. In: Iberian Robotics conference. pp. 537–548. Springer (2019)
13. Teixeira, H., Silva, T., Abreu, M., Reis, L.P.: Humanoid robot kick in motion ability for playing robotic soccer. In: 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). pp. 34–39. IEEE (2020)
14. Xu, Y., Vatankhah, H.: SimSpark: An open source robot simulator developed by the RoboCup community. In: RoboCup 2013: Robot World Cup XVII. pp. 632–639. Springer (2013)