FCYohsin 3D Simulation Soccer Team 2023

Habib Shahzad, Syed Hasan Faaz Abidi, Abuzar Rasool, and Dr. Saleha Raza

Habib University, Karachi, Pakistan

Abstract. FC Yohsin, a team developed by three undergraduate students is a newcomer to the RoboCup 3D Simulation League. The team has developed its own framework in Python to create an autonomous soccer-playing agent capable of competing with other teams in a simulated 3D environment. The paper focuses on the agent architecture, locomotion, and localization of agents in the field and their collaborative decision-making. The team's architecture is modular and consists of sub-modules for perception, decision-making, and action selection. The team has adopted a rule-based approach for decision-making, with different pre-defined heuristics and strategies based on game situations. The paper details the development process of FC Yohsin, including the challenges faced and the strategies employed to overcome them. The team's ultimate goal is to create a competitive team that can navigate the complex soccer environment and outperforms its opponents.

1 Introduction

FC Yohsin is a team that has developed a Robocup 3D simulation team from scratch, using the Python programming language. Under the mentorship of their professor who was part of Karachi Koalas [3], this team intends to create a competitive 3D robot soccer team. Our aim is to not only compete in the RoboCup competition ourselves but to also help newcomers enter this field seamlessly. For this purpose, we have created an open-source framework that provides a high level of abstraction, enabling users to focus on designing strategies and playing styles, while the framework handles the low-level details.

The FC Yohsin team has developed the entire codebase in Python, a versatile and popular programming language, to create a comprehensive framework for developing a RoboCup Soccer Simulation League team. The framework has been designed to be user-friendly and accessible to everyone, and it includes various functionalities such as server communication, parsing of game state data, world modeling, locomotion control, and agent communication in an efficient way.

The team intends to release the framework as an open-source project in near future, allowing new teams to quickly create their own team without worrying about the complexities of the underlying framework. The framework leverages Python's extensive library support, readability, and ease of use to provide a comprehensive solution for creating competitive RoboCup Soccer Simulation League teams.

FC Yohsin's mission is not only to provide a powerful and accessible framework for the RoboCup community but also to be a competitive team. The team

aims to apply the latest research and development procedures to develop better playing styles and strategies that can compete at the highest level in the RoboCup competition. We are constantly exploring new advanced concepts such as reinforcement learning and inverse kinematics to improve our playing style and achieve a higher level of performance. By doing so, the team hopes to set an example for the community and inspire new players to participate in the competition.

2 Agent

The agent architecture was defined in a way such that the agent is able to effectively utilize different components in the system to make decisions on the field. Figure 1 highlights the key components in the agent architecture.

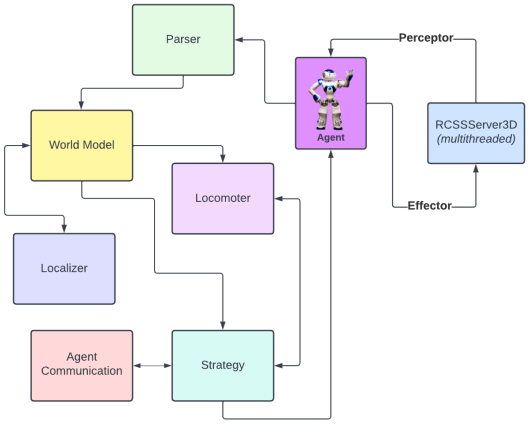

Fig. 1: Agent Architecture

In a soccer game, an agent has several components that work together to help it achieve the collective goal of winning the match. These components include the parser, world model, localizer, locomotor, strategy, and agent communication.

2.1 Parser

The RegEx-based parser is responsible for receiving and parsing data from the server and storing it in the world model. It takes raw data from the server and converts it into a format that the world model can use.

2.2 World Model

The world model is a crucial component of the agent's system that stores the game state of the field from the agent's perspective. It receives parsed data from the parser and adds some necessary information that requires processing with the help of the localizer. The localizer helps the agent to position itself and different objects in the field with respect to the environment.

2.3 Agent Communication

The agent communication module allows agents to communicate with each other via encrypted messages. These encrypted messages are decrypted via the agent communication module and can be used by other modules, such as the strategy module, to inform decisions. The parsed results of these modules are then stored in the world model.

2.4 Localizer

As soon as the parsed data is received in the world model localizer is responsible for converting agents' local coordinates to global coordinates. The localized positions of agents and different world objects are then stored in the World model.

2.5 Locomotor

The locomotor component receives processed information from the world model and helps the agent move specific joints and execute certain skills by simultaneously moving different joints. To perform these actions, the locomotor needs information on the agent's perception of its joints. It is also responsible for reaching a target angle for a specific joint at a specific speed.

2.6 Strategy

The strategy component is responsible for devising a plan or making a decision that aligns with the decision of other agents in the team with the collective goal of winning the soccer match. It interacts with the locomotor component and decides which joints are to be moved and the procedure of moving them.

All the components work together to make a message which is then sent to the server, informing it about the action to be performed. This completes the cycle of the agent's components working together to help the agent achieve its goal in the soccer game.

3 Localization

Localization is a crucial aspect of the RoboCup 3D simulation league, and our team is heavily focused on this aspect to gain an edge over our competitors. To achieve accurate localization of our agents, we use two methods: Triangulation and Particle Filter.

Triangulation is a method that utilizes the known flags and goalposts in the field to estimate the global position of the agent. This method works by measuring the angle and distance between the agent and two or more visible flags. The data coming from the vision perceptor is used to calculate the agent's position and orientation relative to the field. This method is very effective in providing accurate localization when two or more flags are visible. A detailed explanation of this approach is defined in the Section 3.1 of Rezaeipanah et al. (2021) [6]

However, the triangulation method has its limitations. It cannot work if there are fewer than two visible flags or goalposts. This is where the Particle Filter comes in handy. It is a probabilistic method that refines the fluctuating values that triangulation provides. We use it to aid triangulation and as an independent method as well.

The Particle Filter method is a recursive algorithm that estimates the probability distribution of the agent's position and orientation based on the data coming from the vision perceptor. It uses a motion model, which is the planned path of the agent, to predict its future position and orientation. It then compares the predicted position and orientation with the actual values from the vision perceptor to calculate the error. The error is then used to adjust the probability distribution, resulting in a more accurate estimate of the agent's position and orientation.

Particle Filter works particularly well in cases where the triangulation method fails, i.e., there are less than two flags visible. In such cases, Particle Filter is used as a substitute method for localization. Particle Filter takes the planned path of the agent as a motion model for predictions and uses the data from the vision perceptor to calculate the error between the predicted and actual position and orientation. This method provides a good approximation of the agent's location and orientation, allowing us to make effective decisions during the game.

Lastly, for the edge cases where there are no flags visible, we have implemented a default behavior for finding flags by moving the agent. This behavior enables our agents to explore the field and locate the flags on their own, ensuring that we always have some form of localization available.

In conclusion, by focusing on localization, our team has developed effective strategies to navigate the field and compete at the highest level in the RoboCup 3D simulation league. The use of both Triangulation and Particle Filter methods, along with our default behavior for finding flags, ensures that we have a robust localization system in place, allowing us to make quick and accurate decisions during the game.

4 Locomotion

The locomotion skills that are a key component of the agent decision thinking process are defined in this section.

4.1 Walking

The current walk that we have is implemented with the help of a partial series implementation [2]. The paper presents a method for generating a dynamic bipedal walk using partial Fourier series. The method is based on an evolutionary algorithm and uses a set of partial Fourier functions to represent the walking motion. The paper begins by describing the partial fourier series, and how they can be used to represent a variety of different functions. The series is then used to represent the walking motion which is a way of representing a periodic function as a sum of sinusoidal functions. The Fourier Series method was used to represent the walking pattern as a sum of sinusoidal functions and generate the bipedal walk which was evolved and optimized by a genetic algorithm.

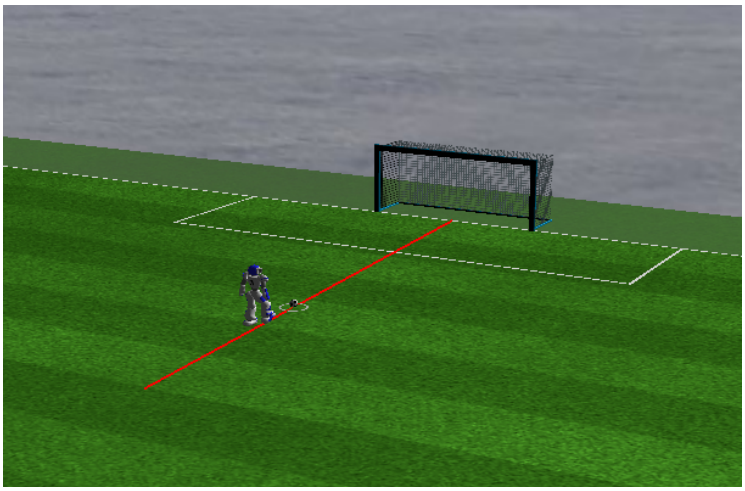

Fig. 2: Agent dribbling the ball towards the goal

In figure 2 we can see how the agent adjusts its desired orientation based on the angle between the goal and itself by attempting to walk in the line.

The drawback of such a walk would be that the nao has limited mobility which makes it difficult to navigate toward a target with ease. Based on the omnidirectional walk engine by UT Austin [4] engine, we are also attempting to develop an omnidirectional walk for our agents, which will provide a significant advantage in executing different strategic skills and movements during the game. This will allow our agents to move in any direction, enabling them to reach their

target positions more efficiently and quickly. With the implementation of this walk, we expect our agents to be more agile and better equipped to handle complex scenarios on the field.

4.2 Turning

Turning was also inspired by the similar mechanism with which the walk was achieved. By utilizing the partial fourier series we are able to achieve a turning mechanism to periodically turn to the desired orientation by tweaking some of the parameters in the partial series based walk.

4.3 Getting Up

In order to recover from a fall the agent has to go through a number of states. Each state has a *wait time* associated with it and in each state, the target angles for a list of specific joints are set. The parameters (angle values for each joint in each state) for getting up were taken by UT Austin Villa base code [5].

4.4 Additional Skills

The process of executing locomotion skills involves a series of sequential states where the agent attempts to move specific joints to reach the desired target angle. After the wait time for each state has elapsed, the agent may move on to the next state in the sequence. Once all the states have been iterated, the locomotion skill is considered complete and can be reset based on the strategy.

This approach is similar to getting up from a fallen state and can be applied to a variety of skills, such as kicking, waving, and moving the head. By breaking down complex skills into a series of smaller, more manageable states, our agent was able to successfully execute these skills in the simulation environment.

5 Strategy

FCYohsin is a team that relies on static positioning and continuous communication to make strategic decisions during the game. Our custom implementation involves defining static positions for each play mode to ensure that each player knows where to go on the field. Our agents continuously communicate their positions to each other, and this information is stored in the world model. This allows us to make strategic decisions based on the location of each player on the field.

5.1 Attack

Our attack strategy involves the closest player going for the attack when the ball is in the opponent's half. Two supporters stay at the back of the attacker to provide additional support as shown in Figure 3. The attacker performs the

dribble-to-goal skill. In case the attacker fails or falls down, the closest supporting agents take the attacker's goal and try to reach the goal to the ball.

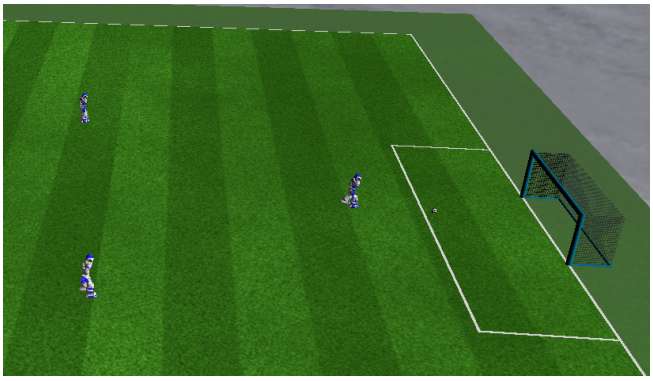

Fig. 3: Attacker with two accompanied attack support agents

5.2 Defense

When the ball is in our team's own half, our defensive strategy is activated. The defenders are accompanied by two other Nao in the penalty box, selected based on their position as shown in Figure 4. The defenders closest to the ball intercept the ball and try to take it beyond the center half. This allows us to effectively defend against our opponents' attacks and prevent them from scoring goals. Currently, we do not have a specific skill for the goalkeeper, and it acts as a defender if the ball enters the penalty box as shown in Figure 5

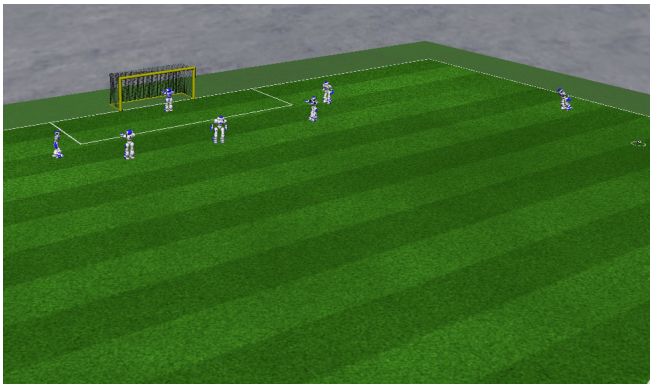

Fig. 4: Defensive mode positioning

While FCYohsin's strategy may not be as advanced as some other teams in the league, it provides us with a solid starting point to think about more complex strategic decisions. Our use of static positioning and communication, combined with our coordinated attack and defense strategies, allows us to compete with other teams and learn from our experiences. We are committed to improving our strategy and performance over time, and we look forward to testing our approach against the best teams in the league.

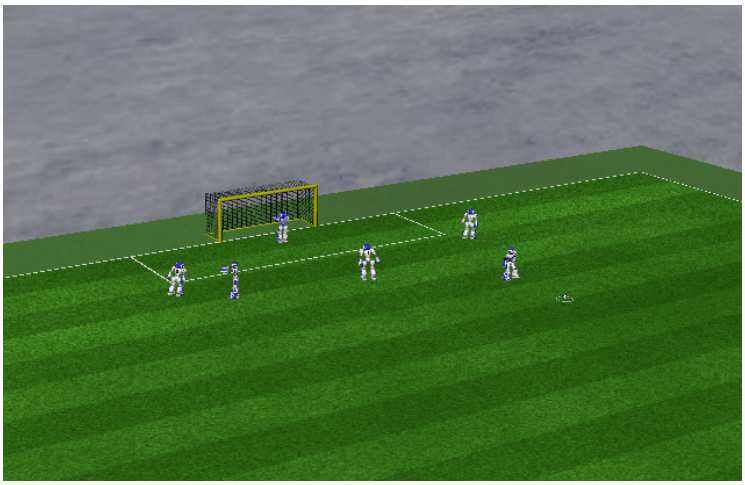

Fig. 5: Intercepting ball in defensive mode

6 Challenges

Our team faced several challenges during the development of our framework for the Robocup 3D simulation league, including:

6.1 Building the framework from scratch using Python

One of the primary challenges we faced was building the entire framework for the Robocup 3D simulation league from scratch using Python. This involved writing every low-level function for communicating and parsing with the server. As Python is a slow language, we had to regularly optimize our base low-level functions to ensure that new skills and strategies do not become performance bottlenecks during the match.

6.2 Parsing the information from the server in an efficient way

Another significant challenge we encountered was parsing the information from the server in an efficient way. Initially, we used loops to go over each segment

of text given by the server, which worked well until we started working on locomotion skills. We began facing performance bottlenecks, and the efficiency of our code deteriorated. We had to rewrite our parser using regex to improve the efficiency of our code.

6.3 Improving the performance of our code

To improve the performance of our code, we employed several strategies:

a. Upgrading from Python 3.7 to 3.11 [7]:
We upgraded our Python version from 3.7 to 3.11, which provided a significant performance advantage. Python 3.11 had improved performance compared to the previous version, which helped us reduce the time required to execute our code.

b. Using libraries like NumPy for faster mathematical computations:
We used NumPy, a popular numerical computing library for Python, for faster mathematical computations. NumPy allowed us to perform complex mathematical operations efficiently and reduced the execution time of our code.

c. Using Cython to optimize the performance of our code:
We used Cython, a programming language that is a superset of Python, to optimize the performance of our code. Cython allowed us to compile our Python code to C code, which provided a significant performance boost.

Despite our code runtime optimizations, we still face some bottlenecks that we are actively working on. Our main goal is to create a solid foundational framework for teams participating in the RoboCup 3D simulation league, enabling them to effectively utilize the most famous programming language used in research, i.e., Python. By doing so, we aim to increase their efficiency and test new skills and strategies faster than before.

7 Future Work

As explained in the section 6, our team is currently facing significant performance bottlenecks when programming agents in Python. In order to overcome these challenges, we are working on optimizing our open-source framework, which we plan to release to the community once we have completed the necessary improvements.

To make our agents more competitive, we plan to explore the use of advanced techniques such as reinforcement learning, inverse kinematics, and evolutionary algorithms. By implementing these techniques, we hope to train our agents to compete more effectively with other teams.

[1] to implement reinforcement learning, which will help our agents learn skills more effectively. This technique involves creating an environment where the agent can learn through trial and error, with rewards given for successful

actions and punishments for unsuccessful ones. By incorporating reinforcement learning into our framework, we hope to significantly improve the performance of our agents.

In summary, our future work involves optimizing our open-source framework and exploring axvanced techniques such as reinforcement learning, inverse kinematics, and evolutionary algorithms. With these improvements, we aim to create more competitive agents and continue to make progress in the RoboCup competition.

References

1. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
2. Haider, S., Abidi, S.R., Williams, M.A.: On evolving a dynamic bipedal walk using partial fourier series. In: 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO). pp. 8–13 (2012). https://doi.org/10.1109/ROBIO.2012.6490935
3. Haider, S., Williams, M.A., Raza, S., Johnston, B., Abidi, S.R., Nafay, A., Rajput, N.: Karachi koalas 3d simulation soccer team team description paper for world robocup 2013 (2013)
4. Macalpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: A winning approach at the robocup 2011 3d simulation competition. Proceedings of the National Conference on Artificial Intelligence **2** (01 2012). https://doi.org/10.1609/aaai.v26i1.8317
5. MacAlpine, P., Stone, P.: UT Austin Villa robocup 3D simulation base code release. In: Behnke, S., Lee, D.D., Sariel, S., Sheh, R. (eds.) RoboCup 2016: Robot Soccer World Cup XX, pp. 135–43. Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin (2017)
6. Rezaeipanah, A., Amiri, P., Jafari, S.: Performing the kick during walking for robocup 3d soccer simulation league using reinforcement learning algorithm. International Journal of Social Robotics **13** (09 2021). https://doi.org/10.1007/s12369-020-00712-2
7. Salgado, P.G.: What's New In Python 3.11 - Python 3 Reference Manual (2023)