Multi-Agent System for Autonomous Table Football: A Winning Strategy

Marcel Založnik*
Jožef Stefan Institute
Ljubljana, Slovenia
marcel.zaloznik@gmail.com

Kristjan Šoln*
Faculty of Electrical Engineering, University of Ljubljana
Ljubljana, Slovenia
ks4835@student.uni-lj.si

Abstract

This paper presents a multi-agent system (MAS) for autonomous table football, developed for the FuzbAI competition at the University of Ljubljana. Our system consists of four independent agents, each dynamically assigned specific roles—Goalkeeper, Defender, Midfielder, and Attacker—based on real-time game analysis. This role-based architecture enabled seamless coordination between offensive and defensive strategies, allowing our team to secure first place. We describe the simulation framework used, the processing of sensor data, and the control strategies that allowed the agents to execute precise actions in a dynamic environment. The results highlight the effectiveness of adaptive, role-based decision-making, demonstrating the potential of MAS in real-time, competitive settings.

Keywords

multi-agent system, autonomous table football, role-based strategy, real-time decision making, AI in robotics

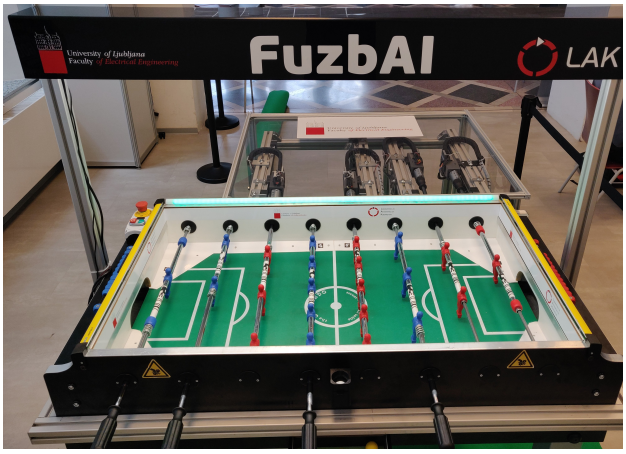

Figure 1: Table setup for the FuzbAI autonomous football competition.

1 Introduction

The FuzbAI competition, held as part of the "Dnevi Avtomatike" event at the Faculty of Electrical Engineering, University of Ljubljana, is a premier contest for students specializing in automation and artificial intelligence [11]. This event challenges participants to develop intelligent autonomous agents capable of playing table football without human intervention. The competition not only serves as a platform for demonstrating technical skills but also fosters innovation in the application of AI and machine learning techniques in real-time environments. Figure 1 illustrates the table setup used in the competition.

The FuzbAI competition is structured in a way that teams must design and implement a fully autonomous system capable of effectively competing against other AI-driven systems. Each match is a test of the participants' ability to integrate advanced algorithms and robotics, simulating the dynamics of a real football game on a miniature scale. The competitive format includes both qualification rounds and knockout stages, ensuring that only the most capable and innovative solutions advance to the final stages.

Our entry into the FuzbAI competition focused on the development of a multi-agent system (MAS), where each of our four rods functioned as an independent agent. These agents were designed to collaborate through a streamlined decision-making process, selecting roles that dictated their actions during gameplay. This strategic approach enabled our team to outperform competitors and ultimately secure first place in the competition.

This paper delves into the development and implementation of our multi-agent system. We will explore the architectural choices, the role-based decision-making strategies employed by each agent, and the overall system's performance in the context of the FuzbAI competition.

2 Competition Setup and System Description

The FuzbAI competition required all participants to develop programs capable of playing table football autonomously. To facilitate this, the competition provided a standardized simulation environment and a set of initial tools that every team used as the foundation for their development. This section describes the simulation framework, the types of data available from the system, and the means by which agents could interact with both the simulated and real game environments.

2.1 Simulation Framework

Participants were provided with a Python-based simulation framework designed to emulate a real table football match, as shown in figure 2. This simulator accurately replicated the physics of the game, including the movement of the ball and rods, and managed the interactions between the environment and the agents controlling the rods. The framework included fundamental functionalities such as ball tracking, rod positioning, and interaction rules, allowing all teams to concentrate on AI development without needing to construct the simulation infrastructure themselves.

One of the key features of the competition setup was that the interaction protocols for the simulator and the physical table were identical. The same signals and commands used to control

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2024.scai.8341

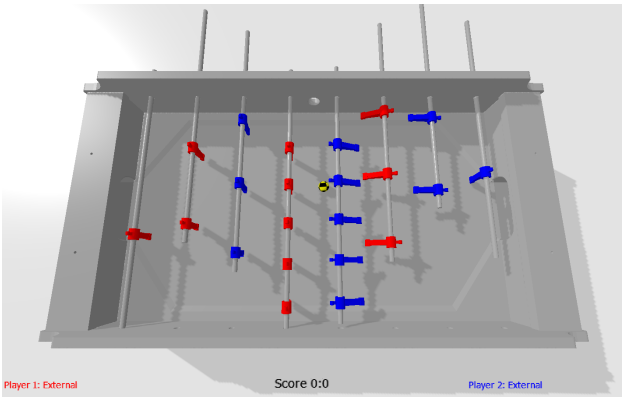

Figure 2: Simulator interface.

the actuators in the simulator were also used for the real table without any modification. This feature ensured that teams could seamlessly transition their algorithms from the simulated environment to the physical table setup, which was used in the final rounds of the competition. As a result, the simulation provided a consistent testing ground that mirrored the actual physical setup, enabling teams to develop and refine their strategies under uniform conditions.

2.2 Sensor Data

Both the simulation environment and the real table provided each team with data from two cameras, one placed on each side of the table. Each camera captured different views of the game, and teams had to decide how to combine the information from both cameras. The data provided by each camera included:

- Ball position: The coordinates of the ball on the 2D plane of the table.
- Ball speed: Velocity of the ball.
- Ball size: Area of the ball in the captured image (in pixels).
- Rod positions: Calibrated position of all rods (in the interval $[0, 1]$).
- Rod angles: Calibrated angle of all rods (in the interval $[-32, +32]$).

This camera data was streamed continuously, requiring teams to process and merge the inputs from both cameras to accurately interpret the game's state. The accuracy and frequency of the data were sufficient to enable real-time decision-making by the autonomous agents, whether interacting with the simulator or the physical table.

2.3 Actuator Outputs

To interact with the environment, each agent could send commands to the actuators that controlled the rods. The system allowed for two primary types of commands:

- Translatory movement: Moving the rod left or right across the table.
- Rotational movement: Rotating the rod to control the angle at which the players struck the ball.

Precise and timely commands were crucial for effective game control, as they enabled the agents to optimally position their figures, strike the ball accurately, and execute defensive or offensive strategies effectively.

3 Related Work

Research on multi-agent systems (MAS) and their application in robotic football has been extensively explored. This section reviews some contributions that have informed the development of autonomous systems for table football and real football.

Moos et al. (2024) [5] developed an automated football table as a research platform for reinforcement learning, highlighting the challenges of transferring learned behaviors from simulation to real-world environments and the need for robust algorithms to handle uncertainties. While reinforcement learning is a common approach in such studies, we did not achieve satisfactory results with it. Therefore, we decided to use multi-agent systems instead. Klančar et al. (2002) [4] investigated cooperative control in robot football (real football) using multi-agent systems, focusing on behavior-based control and dynamic role assignment among robots to optimize performance. Their approach emphasized effective communication for coordination in multi-agent settings. This work particularly inspired our approach to multi-agent systems, where we focused on behavior-based control and dynamic role assignment. Ribeiro et al. (2024) [6] proposed a probability-based strategy (PBS) for robotic football (real football), utilizing real-time data for centralized decision-making without relying heavily on pre-defined plays. Their approach demonstrated flexibility across different environments. Smit et al. (2023) [8] explored scaling multi-agent reinforcement learning (MARL) to a full 11v11 simulated football environment (real football), focusing on computational efficiency and the use of attention mechanisms to enhance scalability in large-scale multi-agent settings. Song et al. (2024) [9] conducted an empirical study on the Google Research Football platform (real football), introducing a population-based MARL training pipeline to quickly develop competitive AI players, highlighting the importance of scalable training frameworks. Scott et al. (2022) [7] examined end-to-end learning in RoboCup simulations (real football), optimizing both low-level skills and high-level strategies through competitive self-play, providing a comprehensive approach to multi-agent training in competitive environments.

4 MAS Approach to Autonomous Table Football Control

In this section, we describe the the methodology of our approach. We describe agent architecture, different agent roles and outline the actions they can take. Then, we discuss the conditions and priorities for role assignment during the game and evaluate the behavior of the system as a whole.

4.1 Agent Architecture

There exist several agent architectures, commonly used in MAS. Approaches, such as [4, 10, 12, 13], use role-based approach for interaction between agents and with the environment. In role-based approach, based on the concepts from role theory [1], the agents are assigned roles which affect their behavior. While the overall long-term goal of the system is typically predefined and does not change, e.g. win a table football match, the current role of an individual agent defines agent's short-term goals, which influences agent behavior, their decision-making process, and how they interact with the rest of the system. Furthermore, separation of agent functionality into independent roles can provide simplification and decoupling of individual tasks, leading to a more modular system, which can simplify and improve the extensibility of the implementation [3].

There exist several approaches to role and behavior implementation in MAS, such as merging different roles, role models and class members [2, 3, 4]. In our implementation, we simplify the architecture by allowing an agent to occupy only a single role at a time, and defining the roles in a way that allows reassigning between iterations of the algorithm without regard to the previous role or state of the agent.

Each role defines a set of possible actions an agent can take. The agents decide which action to take based on their priority and the current environment. More complex roles can be implemented in a stateful manner, meaning the decision on which action to take is dependent on previous actions as well. An agent can only be assigned a single role at a time, but can switch between roles throughout iterations regardless if the particular goal is fulfilled, when appropriate conditions arise. Additionally, every agent must have a role assigned at all times.

An action is a discrete, autonomous task that an agent can take on by making appropriate decisions and acting onto the environment, e.g. by sending commands to the actuators. This advances the agent toward the goal imposed by the current role. An agent can only execute a single action at a time. Additionally, every agent must be actively executing an action at all times.

These concepts were implemented using an Object Oriented approach, as suggested by the authors of the competition. In our implementation, each agent repeatedly executes a fast processing routine. Every iteration, the environment data is updated and role selection for the agent is performed. Then, as the agent decides on a role for that iteration, the appropriate role processing function is called, executing individual actions.

4.2 Role Description

A typical table football setup consists of four rods per player, each with a number of mounted figures. In this implementation, each rod is considered an agent, resulting in a system with four agents for which we define the following roles, typically associated with table football games.

Goalkeeper is the final line of defense, primarily responsible for intercepting the ball before it reaches the goal. Typically the left-most rod, which is nearest to the goal and has a single figure, the goalkeeper follows the ball position using two possible actions: *follow* and *misaligned follow*. The *follow* action simply tries to align the figure on the rod with the current ball position. However, if the velocity of the ball exceeds a predefined threshold, the agent instead attempts to estimate the ball trajectory based on its velocity vector. This estimation is simplified by assuming that the ball maintains a straight-line path. The figure is therefore positioned at the intersection of the rod and the estimated trajectory in an attempt to intercept a fast-moving ball.

The *misaligned follow* action is an augmented variant of the former action, designed to increase the overall defense surface of the defending agents. A common scenario in table football occurs when an attacker attempts to bypass the defenses by slightly pushing the ball parallel to the rod and striking it immediately after. Even though a human player might react fast enough to block such an attack, actuator response times are often insufficient. A defense strategy against such attacks is is to misalign the goalkeeper and defender figures, increasing the defense surface. Here, this is implemented by the *misaligned follow* action, and is activated whenever the ball is relatively slow, in the possession of the opponent and another agent in front of the Goalkeeper is currently in a Defender role. This decreases the chances of the opponent scoring even if the actuators fail to respond fast enough to block this style of attack. Here, communication between the two agents is performed implicitly, as each agent perceives the roles of other agents as a part of the overall environmental state.

Defender is an agent tasked with blocking opponent attacks by intercepting the ball when it is in the opponent's possession or moving towards the goal. This role utilizes a single *follow* action, similar to the Goalkeeper's *follow* action. Whenever the Defender role is active, the agent tracks the position and velocity of the ball, trying to match either its current coordinate or the estimated intersection with the trajectory of the ball. The agent identifies the figure closest to the intersection and attempts to move the rod using minimal amount of movement. This approach allows for faster adjustments during the game, improving defensive efficiency.

Midfielder is a an agent role with the primary task of raising the figures to allow passing the ball from behind the current agent. This role, although simple, is essential in order to avoid accidentally breaking a friendly attack by an Attacker agent behind the current rod.

Attacker is an agent with the task of kicking the ball towards the opponent goal in an attempt to score a point. Unlike other roles, the Attacker role is implemented in a stateful manner. Actions can only happen in a specified order, when the corresponding conditions are met. The role implements *follow*, *kick* and *prevent back-kick* actions.

Whenever the agent is assigned this role, the *follow* action is executed first. During the *follow* action, the agent slightly raises the figures in order to prepare for a kick. The figure closest to the ball is selected and rod offset is adjusted in order to align the figure with the ball. Whenever the agent determines that the alignment with the ball is sufficient, the agent moves onto the next state, the *kick* action. Here, the rod is rotated in order to strike the ball. During this state, it is still necessary to track the position of the ball, as the ball can move significantly within a few iterations of the algorithm. As the rod completes the forward rotation, the agent monitors the position of the ball and assesses if the figure successfully hit the ball. In that case, the next action is set back to *follow*, and the agent is usually assigned a new role according to the environment. However, if the figure missed the ball during the kick, the agent moves onto the *prevent back-kick* action. This final action is meant to prevent an accidental kick in the opposite of the intended direction. The rod is translated sideways and slowly rotated into a neutral position, in order to circumvent the ball. While executing this action, role switching for the current agent is disabled as well.

During execution, the agent aligns the rod position with the ball; however, a perfectly aligned figure results in a straight shot, which is easily defended by maintaining alignment with the ball. A more effective strategy involves kicking at an angle to aim for the goal or create a rebound off the wall, which is harder to defend. This role achieves this by slightly misaligning the figure before and during the kick. The agent computes the angle between the ball's current position and the selected target, with the figure's required misalignment set proportionally to this angle and adjusted by a tunable parameter for fine-tuning. This attack strategy significantly increases the performance of the Attacker role.

4.3 Role Assignment

Individual roles are assigned to agents according to defined assignment conditions and rules. Some approaches use an objective function in order to select a role, often taking role priority into account [4]. In this approach, we instead define a simple set of conditions which, along with role priority, decide on the most appropriate role for a particular agent based on the current state of the environment.

If in a particular instant, more roles fulfill the assignment conditions for a particular agent, the role with higher priority is selected. In this implementation, the highest priority belongs to the Attacker role, followed by the Goalkeeper, Defender and finally the Midfielder with the lowest priority. This ordering is based on the strictness of assignment conditions for each role, and the importance of that particular role. For example, the Attacker role has the strictest selection conditions among all roles, and therefore is assigned the highest priority, while the Midfielder role has a very broad assignment condition and is not as important compared to an Attack agent.

We define the role selection conditions as follows. The Attacker role is selected whenever the ball speed drops below a specified threshold, and the ball is within kicking clearance of the rod. The Goalkeeper role is selected if that particular agent belongs to the left-most rod, closest to the player's goal. The Defender role is selected whenever the ball is in front of the rod. Lastly, the Midfielder role is selected whenever the ball is behind the rod, as the role's only task is to raise the figures to allow the ball to pass forward.

This set of conditions combined with the defined role priority, allow the agents to switch between roles effectively and covers the main functionality required to play the game. Role priority ensures that the agent works toward a correct goal based on the circumstances. For example, any rod, even the Goalkeeper, should attempt to kick the ball if it is close and slow enough, while only the left-most rod should attempt to be the goalkeeper.

4.4 Behavior of the System as a Whole

The system's primary offensive strategy is for the Attacker agents to advance the ball as far forward as possible, ultimately aiming for the goal, while Midfielder agents ensure that they do not obstruct forward passes. During opponent attacks, the systems primary defensive strategy is for the Defender and Goalkeeper roles to intercept the ball. In certain situations, they collaborate to expand the defense surface, compensating for the limitations posed by actuator response times. Once the opponent's attack ends, agents detect the change in the environment and the roles are reassigned to shift the game towards offensive play.

The system's game strategy can be adjusted by modifying parameters such as role priority, assignment rules, or individual actions. For instance, a more defensive strategy can be achieved by tightening the conditions for assigning the Attacker role.

Overall, the implemented algorithm performs well, with the combination of discrete roles resulting in a competent gameplay. However, delays and noise present in measurements, and delays due to actuator response times, sometimes cause the system to miss, e.g. during attacks. The *prevent back-kick* action of the Attacker role proves essential in such situations, performing careful repositioning. Another surprisingly successful strategy is aiming at the goal or the wall during the *attack* action. Even if the ball does not follow the intended trajectory due to measurement noise and system delays, it still considerably increases the attack success rate. Additionally, even though there are no explicit, intentional passes between agents, the strategy of simply passing the ball as far forward as possible is enough for a successful gameplay.

The system overall is sensitive to changes in parameters and requires precise tuning. The simulator, although effective, does not perfectly simulate the physical table, and additional parameter tuning is required when transitioning from the simulator to real-world application.

5 Conclusion

This paper presented a multi-agent system (MAS) for autonomous table football, developed for the FuzbAI competition. Our role-based design allowed each rod to act as an independent agent, dynamically adapting to the game state. This approach enabled effective coordination between offense and defense, contributing to our system's first-place win.

The results demonstrate the effectiveness of a modular, adaptive architecture in dynamic environments, highlighting the importance of robust decision-making and quick role-switching. Future work could include machine learning to predict opponent behavior and optimize strategies, as well as expanding the system to more complex environments. Overall, our MAS showed strong performance in a competitive setting, offering valuable insights for future developments in autonomous systems.

References

[1] Bruce J Biddle. 1986. Recent developments in role theory. *Annual review of sociology*, 12, 1, 67–92.

[2] G. Cabri, L. Ferrari, and L. Leonardi. 2004. Agent role-based collaboration and coordination: a survey about existing approaches. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. Vol. 6. IEEE. ISBN: 1062-922X. DOI: 10.1109/ICSMC.2004.1401064.

[3] E.A. Kendall. 1999. Role modelling for agent system analysis, design, and implementation. In *Proceedings - 1st International Symposium on Agent Systems and Applications and 3rd International Symposium on Mobile Agents, ASA/MA 1999*. IEEE, 204–218. ISBN: 0769503403. DOI: 10.1109/ASAMA.1999.805405.

[4] Gregor Klančar, Marko Lepetič, Boštjan Potočnik, Rihard Karba, and Drago Matko. Cooperative control of mobile agents in soccer game. Faculty of Electrical Engineering, University of Ljubljana, Slovenia, (2002).

[5] Janosch Moos, Cedric Derstroff, Niklas Schröder, and Debora Clever. 2024. Learning to play foosball: system and baselines. In Cornell University Library, arXiv.org. DOI: 10.48550/arxiv.2407.16606.

[6] António Fernando Alcântara Ribeiro, Ana Carolina Coelho Lopes, Tiago Alcântara Ribeiro, Nino Sancho Sampaio Martins Pereira, Gil Teixeira Lopes, and António Fernando Macedo Ribeiro. 2024. Probability-based strategy for a football multi-agent autonomous robot system. *Robotics*, 13, 1. DOI: 10.3390/robotics13010005.

[7] Atom Scott, Keisuke Fujii, and Masaki Onishi. 2022. How does ai play football? an analysis of rl and real-world football strategies. In *International Conference on Agents and Artificial Intelligence*. Vol. 1. Elsevier Scopus, 42–52. ISBN: 2184-3589. DOI: 10.5220/0010844300003116.

[8] Andries Smit, Herman A. Engelbrecht, Willie Brink, and Arnu Pretorius. 2023. Scaling multi-agent reinforcement learning to full 11 versus 11 simulated robotic football. *Autonomous agents and multi-agent systems*, 37, 1. DOI: 10.1007/s10458-023-09603-y.

[9] Yan Song, He Jiang, Zheng Tian, Haifeng Zhang, Yingping Zhang, Jiangcheng Zhu, Zonghong Dai, Weinan Zhang, and Jun Wang. 2024. An empirical study on google research football multi-agent scenarios. *International journal of automation and computing*, 21, 3, 549–570. DOI: 10.1007/s11633-023-1426-8.

[10] Manuela Veloso, Peter Stone, and Kwun Han. 1998. The cmunited-97 robotic soccer team: perception and multiagent control. In *Proceedings of the second international conference on Autonomous agents*, 78–85.

[11] Laboratorij za avtomatiko in kibernetiko. 2024. Dnevi avtomatike. Accessed: 2024-08-21. (2024). https://dnevi-avtomatike.si/?page_id=270.

[12] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. 2003. Developing multiagent systems: the gaia methodology. *ACM transactions on software engineering and methodology*, 12, 3, 317–370. ObjectType-Article-2. DOI: 10.1145/958961.958963.

[13] Xiaoqin Zhang, Haiping Xu, and Bhavesh Shrestha. 2007. An integrated role-based approach for modeling, designing and implementing multi-agent systems. *Journal of the Brazilian Computer Society*, 13, 2, 45–60. DOI: 10.1007/bf03192409.