36th European Modeling & Simulation Symposium, 017
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2024 The Authors.
doi: 10.46354/i3m.2024.emss.017

Digital Twins in Action: Optimizing FuzbAI Through Simulation and Modeling

Matevž Bošnak[1,*], Goran Andonovski[1] and Igor Škrjanc[1]

[1]Faculty of Electrical Engineering, *University of Ljubljana*, Tržaška cesta 25, 1000 Ljubljana, Slovenia

[*]Corresponding author. Email address:
matevz.bosnak@fe.uni-lj.si

Abstract

The FuzbAI system represents a fusion of traditional table football with advanced technology, incorporating computer vision, motion control mechanisms, and artificial intelligence algorithms to provide a new take of the human-machine interaction with the exciting gaming experience. This paper presents a short examination of the FuzbAI system, including its architecture, the development of a digital twin for simulation, and the organization of the FuzbAI competition. Initiated in 2019, the FuzbAI project aimed to attract students to electrical engineering while serving as a platform for testing various AI algorithms. Through the integration of cutting-edge components, FuzbAI offers an immersive gaming experience while preserving the essence of table football. The introduction of a digital twin facilitates rigorous algorithm testing, driving advancements in autonomous gaming. The FuzbAI competition, a showcase of talent and innovation, highlights the potential of AI in gaming. The competition results indicate that sim-to-real approach can be effective tool for simulation-based training and obtaining good performance of AI agents in real-world gaming environments.

Keywords: Table football, Digital twin, Simulation, Modelling

1. Introduction

Robotics and automation in sports is a popular theme, with various endeavors pushing the boundaries of technology and human-machine interaction. From robot soccer, which has been a staple in robotics challenges, to the recent advancements in simulating and training agents using reinforcement learning, the intersection of sports and robotics continues to evolve (Bergkvist and Johansson (2003), Haarnoja et al. (2024)). While robot soccer challenges often focus on relatively slow dynamics and the kinematics of robots, other sports like air hockey present unique challenges due to their fast-paced nature (Ogawa et al. (2011), Liu et al. (2021)).

Table football, a dynamic and popular game, has also attracted attention in the realm of robotics and automation. Efforts such as the "Star Kick" implementation have supported learning, yet lacked a simulation environment (Zhang and Nebel (2007)). Image processing has emerged as a significant challenge in automating table football, as highlighted by previous research (Janssen et al. (2010)). Recent developments have seen attempts at automated table football, with reinforcement learning applied to the kicker using the simulation in Unity 3D (De Blasi et al. (2021)).

Amidst this backdrop, the FuzbAI system represents a convergence of traditional table football with cutting-edge technology. This paper explores the architecture of FuzbAI and the development of its digital twin, designed to replicate the system's operation within a simulated environment. By enabling Sim-To-Real transfer learning, the digital twin facilitates the seamless transition of AI algorithms from simulation to real-world application, a topic explored in recent surveys (Zhao et al. (2020), Salvato et al. (2021)).

Originating in 2019, the FuzbAI project aimed not only to create an engaging promotional project but also to attract students to electrical engineering while serving as a platform for testing various computer vision, control, and AI algorithms. Leveraging state-of-the-art components, including computer vision algorithms and motion control mechanisms, the FuzbAI system delivers an immersive gaming experience against a non-human opponent, all while preserving the essence of traditional table football.

The integration of a digital twin not only allows for rigorous testing and optimization of AI algorithms but also opens avenues for future advancements in autonomous gaming. The FuzbAI competition, an embodiment of talent and innovation, serves as a testament to the potential of AI in gaming and improves students engagement beyond the classical lecture-based interactions. It also serves as a compelling AI research topic, paving the way for further exploration in this exciting field.

While the students in the competition primarily focused on developing and advancing the virtual agent/player to outperform opponents, the system itself can also be utilized to train human operators/players, a well-established practice in the industry, as demonstrated by Baratta et al. (2023).

The paper is structured as follows. First, an overview of the FuzbAI system architecture is provided, followed by a discussion of the modeling and simulation environments. The paper concludes with a focus on the FuzbAI competition event, detailing its organization and outcomes.

2. FuzbAI system

The FuzbAI system (illustrated in Fig 1) is based on the standard competition-level football table manufactured by Garlando. The initial idea of supporting various type of human-machine game configurations led us to avoid modifying any critical part of the table that is used for the game-play. All play rods were therefore left intact and no modification was done to the playground surface. A pair of Basler acA1440-220uc cameras with 4 mm lenses is mounted over the table with good separating distance between them. Such positioning of the cameras allows reliable ball position determination even in case of obstructions (ball under the play rod or the player). Our bespoke ball tracking algorithm uses target object mask correlation technique to determine the center of the ball even in cases that only a part of the ball outline is visible by the camera. The vision system also enables accurate determination of play rod positions and rotations via the use of visual encoders, as described in Bošnak and Klančar (2020).

In order to support mounting of actuators to the side of the table, a frame made of standard 45 mm aluminum extrusion profiles was first affixed to the table. Modular drive units are then mounted to this frame where needed with the most commonly used configuration of having all 4 playing rods on one side (e.g. red player) driven with these, as shown in the Figure 1 and allow the game of type *AI (red) vs. human (blue)*. Each 2-DoF modular drive unit consists of two BLDC (brushless DC) motors, one actuating the linear axis and the second motor actuating the rotational axis. Both motors are equipped with incremental encoders and controlled in positional closed loop using the ODrive motor controller. Motion of all drive units is synchronized in the motion controller card PoKeys57CNC, which takes care of enforcing the limits for the motion dynamics by limiting the dynamics of the reference position for the motor drivers (limiting maximum acceleration and velocity). Separate safety system uses light curtains that cover the area above the playground and trigger the emergency stop in the motion controller in any object is detected crossing the safety plane.

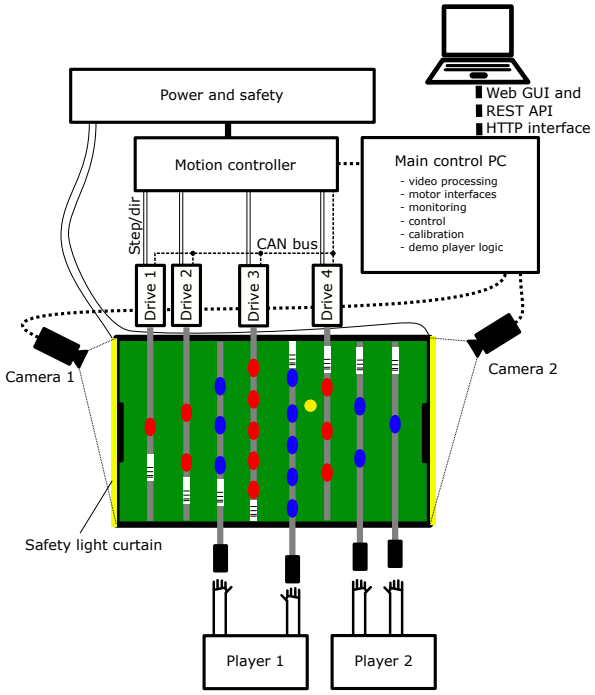

Figure 1. FuzbAI system diagram illustrates major components and their interactions. A pair of human players (at the bottom) control the rods with blue figurines, while the motion control system (on the top) controls the red ones. Cameras with additional lights are positioned above the playing field, observing the table from two different angles, while the rest of the system is neatly hidden below the table.

The control logic is implemented on the desktop PC running Linux operating system. The software is divided into computer vision processing application, written in C++, that handles the ball tracking and decoding of playing rods position and rotation. The main module is implemented as an ASP.NET application that serves as an interface between the FuzbAI system and the user. It handles system initialization procedures, implements HTTP REST API data exchange protocol and serves user interfaces for control, calibration and rendering. The game-play logic (agent) is implemented as a separate process that uses the REST API for data exchange, both for reading the state of the system (ball position, play rod positions) and controlling the motors. An agent for a game-play demonstration is implemented on the same desktop PC, but can also be im-

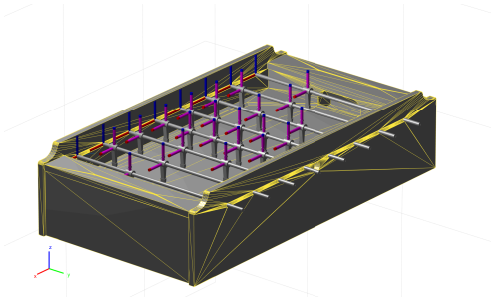

Figure 2. URDF-based 3-D model of the system, also showing the wire-frame of the table model and the joints, specified in the URDF tree.

plemented externally.

3. Modelling the FuzbAI system

The FuzbAI system was modelled for the purpose of designing a digital twin that would enable testing and experimentation with the agent logic. From the agent's point of view (REST API), the digital twin should behave the same as the real system. This requires modelling of the actual game environment, physics, actuator dynamics and other intermediate logic layers.

3.1. 3-D model and simulation environment

Table football game is based on mostly elastic collisions of the ball with the environment and the player figurines. First tries of modelling and ad-hoc simulating the game quickly proved to be a very demanding task. Although, the game appears to be fully encased in a two-dimensional domain, all collisions must be evaluated in a three-dimensional space. For the purpose of simulation, different environments have been evaluated (a list of different physics engines is also provided in Zhao et al. (2020)), finally deciding for an open-source physics engine Bullet (presented in Coumans (2005)). PyBullet (Python-wrapper for Bullet library) was used for the simulation of the game-play in real-time (rendering window shown in Figure 3).

The 3-D model of the FuzbAI system was carefully constructed in the Fusion360 CAD tool. The model contains accurate representation of the actual table, including the ball ramps on the sides of the field. The playing rods with the player figurines were modelled as separate entities. Meshes of all objects were exported into standard STL file format and then joined using the URDF (Unified Robot Description Format) file (as shown in Figure 2). URDF is a standardized file format for representing mostly robot models, employing tree-like structure that describes different robot parts and the joints between them. The combination of prismatic and revolute joints were used to represent the axially-constrained mechanics of the playing rods.

Since the Bullet physics simulator does not support dynamic (in terms of collision detection) concave objects and

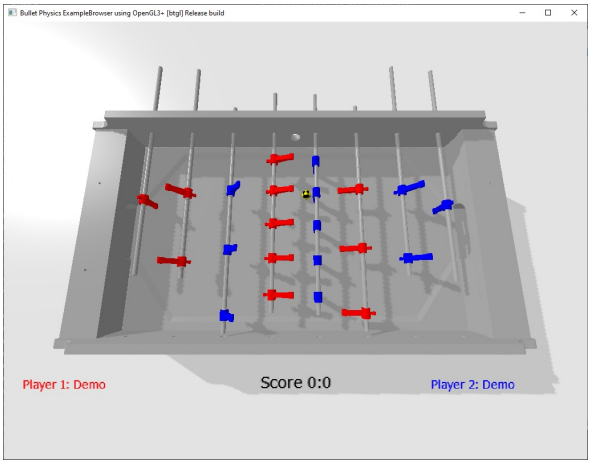

Figure 3. FuzbAI simulator in Bullet physics engine renders the current state of the simulator allowing the game to be monitored in real-time

the model of the table was concave (as shown in Figure 2), the table without play rods was first created as a static solid body in the simulation and URDF was loaded afterwards, importing other objects (such as play rods and player figurines). The triangle count (indicating the object's complexity in terms of collision events evaluation) was kept as low as possible, but the models of individual parts were shared for both the visual rendering and physics engine. The actual table football ball was represented as a spherical object with customized dynamics parameters (lateral, rolling and spinning friction, restitution and damping), that were manually tuned by launching a set of experiments in simulation and real-world environments.

The actuators were modelled with the joint motor controllers, supported by the Bullet engine. Since these enable the maximum torque/force and velocity to be specified, the values for those were chosen to result in similar dynamic response of an actual system. The scaling of the control signals was chosen to be identical to the signals on the real device. Additionally, the control signal nonlinearity of type dead-band was implemented to account for a similar filter implemented on the real system.

Sensor system of the FuzbAI system is mostly comprised of video cameras and computer vision algorithms – since the position of the ball and the play rods is always known in the simulation, we mostly dealt with determining what level of noise needs to be added to the simulated data to represent the real-world system. Uniformly distributed random noise was added to the identified ball position and velocity in x-y plane and additional positional shift was added as a result of ball's z-position above the play surface (in case the ball jumps on collisions).

Since the inherent delays observed in the actual FuzbAI system (due to data transfers, image processing, etc.) could be regarded as a single delay block in the closed-loop system, these were not individually modelled as a part of an actual physics engine, but were added to the output data stream (between the simulation and the user).

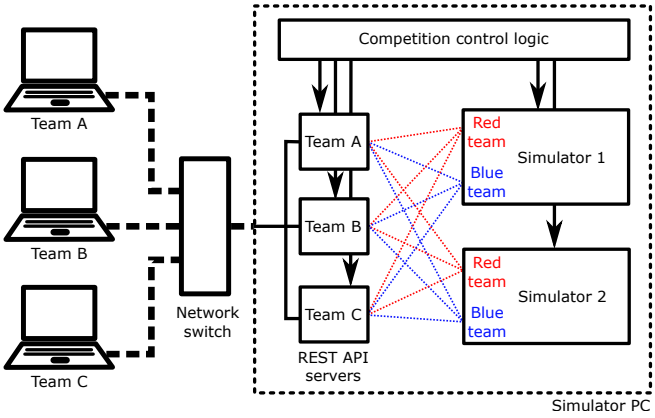

Figure 4. Illustration of the FuzbAI simulated competition system, showing all major sub-systems. The right part of the system is implemented as separate processes on the simulator PC and is decorated with a dotted square.

3.2. Simulated REST server

Since PyBullet running in realtime imposes stringent timing requirements due to the simulation loop and the fact that Python process can only concurrently run a single thread, the logic of the REST API server was moved to a separate Python process and implemented using the FastAPI and uvicorn Python libraries. Communication between the simulator and the server processes was implemented as a socket-based IPC (Inter-Process Communication) based on UDP. This separation allowed us to split the agent and simulator interface implementations (as illustrated in Figure 4), allowing individual agent interface (REST API server) for each of the team. Since each of those interfaces mimicked the operation of the REST API server on the FuzbAI system, agents could not discern the nature of the opponent (real human in case of FuzbAI system or another agent in case of the simulation). Additionally, the described approach enabled concurrent running of 4 simulators on a standard desktop PC, each running in its own thread.

3.2.1. Competition control logic

Since the communication channels between servers and simulators is re-configurable on-the-fly, the competition control logic was implemented to assign each team to specific playing position in the simulation. This facilitated support for multiple simulators to be running concurrently, where mapping of teams to red/blue positions and games was configured based on the competition requirements.

Since the agents in the original FuzbAI system always control the red player rods, the competition control logic needs to augment the blue team's players (both the controls and measurements) so that the agent logic perceives them as the red team. Thus, each simulated REST server can connect the team's agent to any team in the simulators, automatically converting the data sent and received accordingly.

The competition rules limited the time of each match to 2 minutes or the moment first team reaches score of 5 goals. This was enforced by the competition control logic, which was able to reset the state of each simulator, start and stop the gameplay. Competition manager (as shown in Figure 4), which implemented this logic in the form of another Python-based server, also served the GUI for the operator.

4. FuzbAI competition

The Laboratory of Control Systems and Cybnernetics at the Faculty of Electrical Engineering, University of Ljubljana hosted the FuzbAI competition for the first time for students of the university. There were 8 team applications and 5 of those competed at the actual event.

Each team competed with both the human players and their agent trying to outperform the other teams. The competition had tournament nature – first, qualification round matches were played in the simulator against team agent's in order to determine the team pairs in the tournament rounds. The tournament round matches were played on the real FuzbAI system, where each pair played two games, first between team A human players and team B agent then the roles were reversed for the second game, where team A agent played against team B human players (see Table 1). The winner of the match was determined by combining the results of both games and the result of the qualification round match between the two teams involved. The result of combining the results of qualification and tournament games was increased focus on the agent's performance and less on the actual human player performance.

4.1. Automatic referee

Although, the qualification games had no referee system implemented and relied only on detecting stalled balls and relaunching them into the game, the tournament games involved human players and the organizers wanted to prevent (or at least reduce) the possibilities for unfair play. The main rule enforced by the automatic referee was a limit on the ball posession time (similar rules exist for the table football competitions) in order to facilitate dynamic play. A special care had to be taken since the Garlando table used in the FuzbAI has dead zones, areas on the play-field, where the ball is out of reach of all players. Automatic referee accounted for this zones and automatically called a foul if ball posession timeout was reached or if ball was detected as being stalled in the dead zone.

4.2. Teams, training, approaches

As noted, there were initially 8 student teams that applied for the competition. All teams were given access to the FuzbAI simulator, which allowed them to work on their algorithms. There were no conditions given to the teams regarding the nature of the implementation (except for it

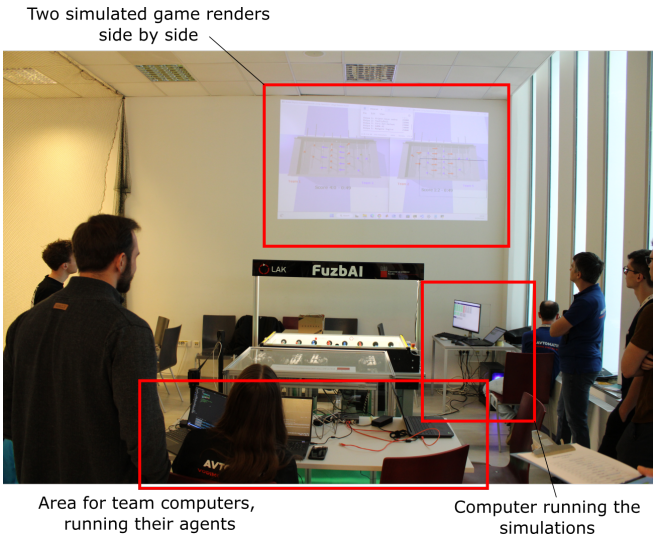

Figure 5. Picture taken during the qualifications, where all competing teams connected their computers to the network switch and had their agents play other teams in one of the two simulated sessions.

requiring no human intervention and using REST API for communication). Initially, about half of the teams started by using simulator as a gym environment for the machine learning, while other team decided for an expert system approach, where they implemented rules for motion directly in the code. Although, the simulator was made available 1 month before the competition, the complex nature of the problem resulted in AI training sessions not indicating convergence towards the desired results. As the result of that, all 5 teams that actually participated in the competition, used expert system approach for their agents (similar conclusions as in Pepelnjak (2023)).

Besides training on a simulator, teams were given access to the FuzbAI system for testing 1 week before the competition. As a sidenote, the team that finished second overall, attended no training on the FuzbAI system and tested their agent solely in the simulated environment provided.

4.3. Competition and results

Competition was held as a part of the event *Dnevi avtomatike 2024* (Days of Automation 2024) in parallel with the drone competion on April 10th, 2024 in the lobby of the Faculty of Electrical Engineering, University of Ljubljana. The competition opened with the qualifications event (Figure 5), where all teams connected their computers to a network switch and accessed the simulated environment with their own dedicated REST API server (each team server was started on a different port). During qualifications, each team played a match of two games against all other teams. Results are shown in third column of the Table 2.

Due to the number of teams, only one quarter-final, two semi-final and two final matches was played (results of all matches shown in Table 1, while a picture taken during one of the games is shown in Figure 6). Looking at the

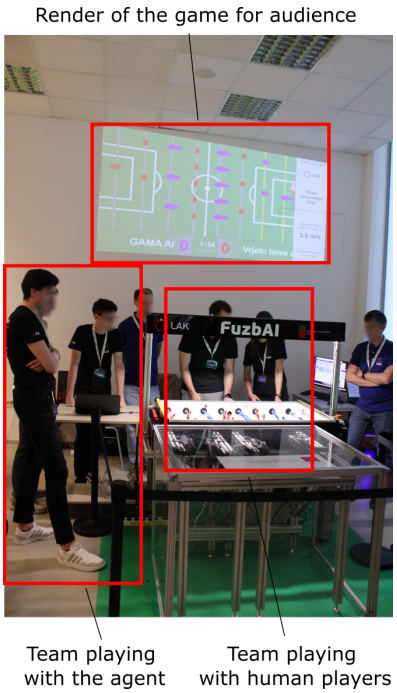

Figure 6. Picture taken during the tournament, where only two teams competed at a time, one playing with their agent against humans of the second team.

results, one can discover that human players have beaten agent opponents in 6 of the 10 games played, tied in 3 and lost 1 game. Although the qualification advantage of Team C helped the team win the first match, the qualification results did not practically effect the other games. Moreover, the team standings after the qualification are practically equal to the final results (see columns 1 and 3 of the Table 2). The later can be seen as a way to estimate the simulator performance indicator – the agent that performed well in the simulation (e.g. agent of Team A), also performed better in the game against the human (agent of Team A played one tie and lost once). Vice versa, agent that performed worst in the simulator (Team C), performed worst also in the tournament against human opponent (agent of Team C lost all games against human).

Besides the results, agent performance can also be analyzed via the data saved during the games. Unfortunately, the simulation system did not produce logs during the qualifications and only logged data of the tournament games exist (shown in Figure 7). Logs indicate that agent of the Team A performed really well defense positions, preventing the human opponents from scoring a goal easily. Logs also indicate that human players of Team A were very successful in scoring goals against agents of teams C and D, proving that the winner of the competition indeed had the best agent and human players overall.

5. Conclusion

Training of agents in simulation is a successful and well-established approach, which was also employed during the

Table 1. Results of the tournament games, where each team competed against other teams with both their agent and human players in two matches (human players of team 1 versus agent of team 2 in the first match and agent of team 1 versus human players of team 2 in the second match.

Round	Team 1	Team 2	Qualif. adv.	Human 1 vs. Agent 2	Agent 1 vs. Human 2	Final score	Winner
1/4-final	Team C	Team E	1.5 : 0	0 : 0	0 : 1	1.5 : 1	Team C
1/2-final	Team A	Team C	4 : 0	5 : 0	1 : 3	10 : 3	Team A
1/2-final	Team B	Team D	0 : 0.5	2 : 3	1 : 1	3 : 4.5	Team D
3rd place	Team C	Team B	0 : 1.5	4 : 3	1 : 3	5 : 7.5	Team B
Finals	Team A	Team D	5 : 0	5 : 1	2 : 2	12 : 3	Team A

Table 2. Final results overview, showing a clean victory for team A with best performing agent and human players.

Place	Team	Qualification result	Tournament - agent	Tournament - human	Tournament - team
		Won / Tied / Lost - Score	W/T/L	W/T/L	W/T/L
1	Team A	8 / 0 / 0 - 24 (1st place)	0 / 1 / 1	2 / 0 / 0	2 / 1 / 1
2	Team D	3 / 1 / 4 - 10 (2nd place)	0 / 0 / 2	0 / 2 / 0	1 / 2 / 2
3	Team B	2 / 3 / 3 - 9 (3rd place)	0 / 1 / 1	0 / 0 / 2	0 / 1 / 2
4	Team C	2 / 0 / 6 - 6 (5th place)	0 / 0 / 3	1 / 1 / 1	1 / 1 / 4
5	Team E	2 / 2 / 4 - 8 (4th place)	0 / 1 / 0	0 / 0 / 1	0 / 1 / 1

presented FuzbAI competition. The competing teams reported no issues or agent performance degradation when transitioning from simulation to the real world, which solidifies the use of simulation in such applications. Moreover, some of the teams used solely simulated environment for testing and training of the agents, which performed on the real device as well as in the simulator, strengthening the applicability of the Sim-To-Real transfer approaches.

Based on the nature of the problem to be solved in the competition, we expected some of the teams to enter the competition with a solution that is the product of reinforcement learning approach, but no team was able to produce such viable solution in time for the competition. Instead, all solutions were based on the expert system approach, where the expert knowledge was implemented in code to control the agents. We expect that the solution based on machine learning algorithms would outperform such approaches and take advantage of simulation-based training sessions in producing motion sequences that are hard (or even impossible) to implement as a state-machine based code. However, highly-dynamic interaction of agents with the environment makes the problem space more complex than it appears to be on the surface and the far greater amount of simulated training would be required.

We plan to host the competition again in the following years and will work on improving the simulator environment in order to better facilitate machine learning. We plan to implement changes to the simulator that would allow faster than real-time execution and produce a gym-like environment with multiple entities simultaneously. The current version of the simulator is available at https://github.com/FE-LAK/FuzbAISim.

Improvements and upgrades to the system's mechanical design are also planned to increase the achievable linear acceleration and velocity capabilities, which are the main limitations of the current system. A comprehensive analysis of the computer vision system is also in the making with the focus on the actual timing of an image capture, transfer and processing processes. Accurate estimation of delays in these systems will allow for improved delay compensation techniques.

References

Baratta, A., Cimino, A., Gnoni, M. G., Mirabelli, G., Padovano, A., and Talarico, S. (2023). How cognitive capabilities for smart operator enhance human-robot collaboration. In *Proceedings of the 35th European Modeling Simulation Symposium (EMSS 2023), 049.*, volume 2023-September.

Bergkvist, M. and Johansson, S. (2003). *Machine learning in simulated RoboCup Optimizing the decisions of an Electric Field agent*. Master thesis, Blekinge Institute of Technology.

Bošnak, M. and Klančar, G. (2020). Fast and Reliable Alternative to Encoder-Based Measurements of Multiple 2-DOF Rotary-Linear Transformable Objects Using a Network of Image Sensors with Application to Table Football. *Sensors*, 20(12):3552.

Coumans, E. (2005). Bullet Physics Engine.

De Blasi, S., Klöser, S., Müller, A., Reuben, R., Sturm, F., and Zerrer, T. (2021). KIcker: An Industrial Drive and Control Foosball System automated with Deep Reinforcement Learning. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 102(1).

Haarnoja, T., Moran, B., Lever, G., Huang, S. H., Tirumala, D., Humplik, J., Wulfmeier, M., Tunyasuvunakool, S., Siegel, N. Y., Hafner, R., Bloesch, M., Hartikainen, K., Byravan, A., Hasenclever, L., Tassa, Y., Sadeghi, F., Batchelor, N., Casarini, F., Saliceti, S., Game, C., Sreendra, N., Patel, K., Gwira, M., Huber, A., Hurley, N., Nori, F., Hadsell, R., and Heess, N. (2024). Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022.

Janssen, R., De Best, J., Van De Molengraft, R., and Steinbuch, M. (2010). The design of a semi-automated football table. *Proceedings of the IEEE International Conference on Control Applications*, pages 89–94.

Liu, P., Tateo, D., Bou-Ammar, H., and Peters, J. (2021). Efficient and Reactive Planning for High Speed Robot Air Hockey. *IEEE International Conference on Intelligent Robots and Systems*, pages 586–593.

Ogawa, M., Shimizu, S., Kadogawa, T., Hashizume, T., Ku-

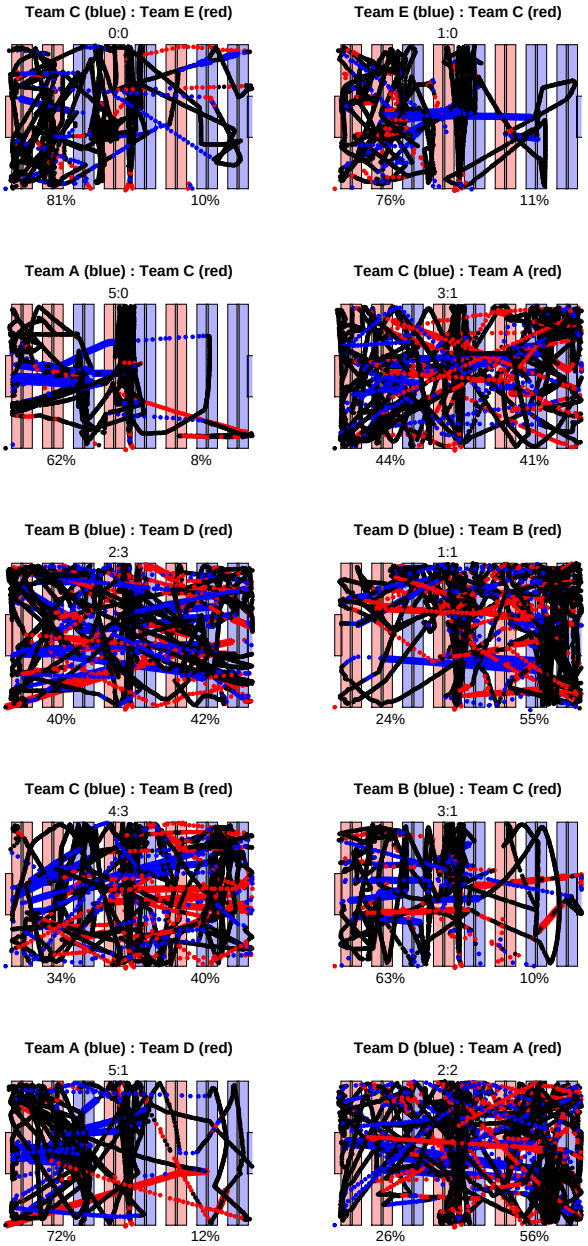

Figure 7. Logged data of the tournament games - blue opponent is always a human, red is an agent. Blue color in trajectories indicate ball moving towards the red goal (left), while red color indicate ball moving towards blue goal (right). Score of the game is shown above each graph, ball possession is shown below the graphs, on the left for red player, right for blue player.

doh, S., Suehiro, T., Sato, Y., and Ikeuchi, K. (2011). Development of Air Hockey Robot improving with the human players - Arm control for effective quick draw taking an opponent-s eye movement into account-. *IECON Proceedings (Industrial Electronics Conference)*, pages 3364–3369.

Pepelnjak, L. (2023). *Implementation of Table Football in Virtual Reality*. Batchelor thesis, University of Ljubljana.

Salvato, E., Fenu, G., Medvet, E., and Pellegrino, F. A. (2021). Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187.

Zhang, D. and Nebel, B. (2007). Learning a table soccer robot a new action sequence by observing and imitating. *Proceedings of the 3rd Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2007*, (January 2007):61–66.

Zhao, W., Queralta, J. P., and Westerlund, T. (2020). Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pages 737–744.