FRA-UNIted — Team Description 2025

Thomas Gabel, Berkan Eren, Jorge Vanegas, Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de, {berkan.eren|vanegas}@stud.fra-uas.de

Abstract. The main focus of FRA-UNIted's effort in the RoboCup soccer simulation 2D domain is to apply machine learning techniques in complex domains. In particular, we are interested in applying reinforcement learning methods, where the training signal is only given in terms of success or failure. In this paper, we review some of our recent efforts taken during the past year, putting a special focus on extension to the physical soccer simulation model and teams' computational demands.

1 Introduction

The soccer simulation 2D team FRA-UNIted is the continuation of the former Brainstormers project which has ceased to be active in 2010. The ancestor project was established in 1998 by Martin Riedmiller, starting off with a 2D team which had been led by the first author of this team description paper since 2005. Our efforts in the RoboCup domain have been accompanied by the achievement of several successes such as multiple world champion and world vice champion titles as well as victories at numerous local tournaments. Our team was re-established in 2015 at the first author's new affiliation, Frankfurt University of Applied Sciences, reflecting this relocation with the team's new name FRA-UNIted.

The underlying and encouraging research goal of FRA-UNIted is to exploit artificial intelligence and machine learning techniques wherever possible. Particularly, the successful employment of reinforcement learning (RL, [9]) methods for various elements of FRA-UNIted's decision making modules – and their integration into the competition team – has been our main focus. Moreover, the extended use of the FRA-UNIted framework in the context of university teaching has moved into our special focus.

In this team description paper, we refrain from presenting approaches and ideas we already explained in team description papers of the previous years [3, 4]. Instead, we focus on topics that have moved into our focus in the course of the past year. As you will see, over the past year, the *tackle command* has increasingly become a focus of our interest. We start this team description paper, however, with a short general overview of the FRA-UNIted framework. Note that, to this end, there is some overlap with our older team description papers including those written in the context of our ancestor project (Brainstormers 2D, 2005–2010) which is why the interested reader is also referred to those publications, e.g. to [8, 7].

1.1 Design Principles

FRA-UNIted relies on the following basic principles:

- There are two main modules: the world module and decision making
- Input to the decision module is the approximate, complete world state as provided by the soccer simulation environment.
- The soccer environment is modeled as a Markovian Decision Process (MDP).
- Decision making is organized in complex and less complex behaviors where the more complex ones can easily utilize the less complex ones.
- A large part of the behaviors is learned by reinforcement learning methods.
- Modern AI methods are applied wherever possible and useful (e.g. particle filters are used for improved self localization).

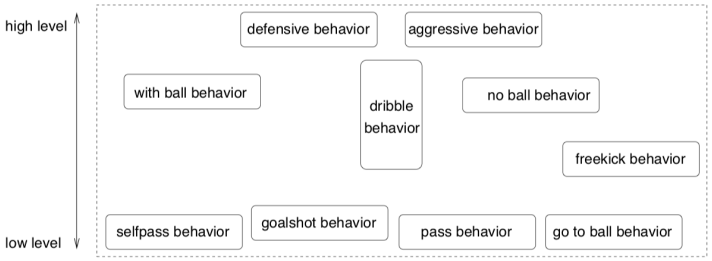

Fig. 1. The Behavior Architecture

1.2 The FRA-UNIted Agent

The decision-making process of the FRA-UNIted agent is inspired by behavior-based robot architectures. A set of more or less complex behaviors realize the agents' decision making as sketched in Fig. 1. To a certain degree this architecture can be characterized as hierarchical, differing from more complex behaviors, such as "no ball behavior", to very basic, skill-like ones, e.g. "pass behavior". There is no strict hierarchical sub-divisioning, which is why a low-level behavior may call a more abstract one. For instance, the behavior responsible for intercepting the ball may, under certain circumstances, decide that it is better to not intercept the ball, but to focus on more defensive tasks and, in doing so, call the "defensive behavior" and delegating responsibility for action choice to it.

2 Bipedal Running Model and Evaluation

Over the past year, we have expanded the FRA-UNIted infrastructure, focusing particularly on refining motion control. Our objective was to react to the introduction of the bipedal running model with version 19 of the Soccer Server [6]

making it usable for our agents and evaluating its effectiveness. This enhancement aimed to optimize movement sequences, making them both more realistic and more efficient. The integration of this system required various modifications to our existing software architecture, particularly in terms of class structures and control methods.

Software Adjustments: To successfully integrate the bipedal running model, a couple of changes were made to key components of the FRA-UNIted agent, including classes for representing action commands and for realizing the communication with the Soccer Server. Internally, a new action type, `TYPE_DASHLR` was introduced, which allows for the independent control of both legs. In addition to these structural updates, new control methods were developed to facilitate bipedal running. These adjustments ensure that movement commands are processed efficiently while maintaining compatibility with existing system components. Beyond these updates, refinements were also made to tools classes, particularly enhancing a modelling method, which allows us to simulate the movement that some command will yield, to also cover bipedal movements.

Use Case "Tacklish" Intercept: A first application of this new movement model is showcased in a behavior for ball interception with maximized tackle success probability called `TaklishIntercept2024`. Using this behavior, a player attempts to tackle a rapidly moving ball, which it will just barely fail to intercept. To address this challenge, an optimized interception strategy was developed, leveraging the new bipedal movement capabilities. It strives for nearly reaching the ball, bringing it into tackle range with a greater tackle success probability. If conditions allow, one or more final `TYPE_DASHLR` command(s) is/are executed at the last possible moments to maximize the chances of a successful tackling. The tacklish interception point itself is precisely calculated maximizing the probability of a successful final tackle. Extensive simulations of this scenario demonstrated an improvement in tacklish interception success rates, highlighting a first practical benefit of the newly implemented bipedal running model.

3 Holes and Clashes in Simulated Soccer

An interesting new line of research within our team was sparked after the RoboCup 2022 tournament, where we witnessed particularly peculiar scene during the match against team Tehran. Using flip book-style, Fig. 2 visualizes this situation in which our team's agent #2 (blue) performed a failing tackle at $t = 4792$ in a state where the tackle success probability was zero, whereas it would have been quite promising to perform that tackle a time step earlier, i.e. at $t = 4791$ where its chance of success would have been 84.6%.

Knowing that our team's calculation for the tackle decision to perform a tackle is efficient and fail-safe, the observed erroneous behavior seemed dreadful and miraculous. Inspecting the game's logfile revealed, however, that #2 apparently intended to perform its tackling at $t = 4791$, but that the corresponding

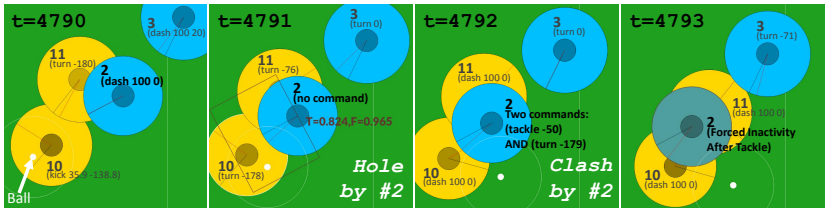

Fig. 2. Tracking the Sequence of Actions of Player #2: After a dash with 100% power into direction 0°, the rectangle inside player #2 at $t = 4791$ indicates a tackle success probability of $T = 0.824$. Instead of doing the tackle, however, #2 issues no command at all – a hole arises. In the subsequent cycle $t = 4792$, two commands from #2 arrive at the server – a clash. While the latter command is ignored by the server, the former one, a tackle into direction -50°, is executed. Though this would have been a good action at $t = 4791$, it is doomed to fail at $t = 4792$ since now the ball is already so distant that the tackle success probability is zero. After the tackle has failed, #2 is, according to the rules of the simulation, forced to 10 steps of inactivity starting at $t = 4793$ which allow the opponent players #10 and #11 to dribble ahead.

command *arrived late*, i.e. not until the succeeding time step, followed by a turn command which was intended for $t = 4792$. That situation is called a *hole* (no command at $t = 4791$) with a subsequent *clash* of two commands, (which is sometimes cordially denoted as a "black hole"), where the soccer simulator executes only the first of the commands arrived and disregards the latter.

The rest is history: Since the situation took place near our team's penalty area and having agent #2 forced to ten cycles of inactivity after its failed tackle, the opponent team scored a goal. Challenged by this instance, we decided to research on the topic of holes and clashes in soccer simulation 2D in more depth. We here provide an excerpt of our findings [5], starting off by introducing some necessary notation. In Section 3.2, we then present a *historical* review on the occurrence of holes and clashes covering the entire history of the RoboCup initiative.

3.1 Notation

In the early days of RoboCup's soccer simulation, around the turn of the millennium, it was a challenge to assure synchronization between the soccer-playing agents and the Soccer Server. On the one hand, the UDP protocol used for exchanging messages did not guarantee the arrival of some packet at the target implying that the agents' states were eventually not synchronized with the server cycle. On the other hand, games are run under real-time constraints and, depending on the computers (their particular hardware characteristics and computing performance) that were employed at a some specific RoboCup tournament's venue, the agents' computational thirst at times exceeded the available resources often resulting in messages to arrive too late at the server's side.

Definition of Holes: In its nowadays used setup, each agent is allowed to send a single body-related command to the Soccer Server in each time step which

may be a dash, turn, kick, tackle, or catch (goalie only) action. We say that a *Hole* is a situation where no such body-related command has arrived at the Soccer Server from an agent. Thus, there can be up to eleven holes per team per step. And given that the default duration of a match is ten minutes (i.e. 6000 simulation cycles), there might be up to 66.000 holes per team per match. Note that, to this end, we focus on the players only and disregard the coach agent and its capabilities to provide advice to the players.

Generally, the occurrence of a hole *may* indicate a potential problem, but it does not necessarily have to. While a hole certainly represents an agent's missed opportunity to act, there might also be times where it intentionally sends no command, if, for example, it sees no need to act.

Definition of Clashes: A *Clash* denotes the situation where the Soccer Server receives more than only one body-related command within one simulation cycle from one specific agent. Unreliable network connections and variations in the speed with which a network packet containing a command is transmitted used to be an erstwhile cause for clashes. Today's main cause for clashes are compute-hungry agents or, stated differently, underpowered simulation machines. If an agent's calculations for determining its next action takes longer than 100ms and, thus, then the command cannot arrive at the Soccer Server within the current simulation cycle, but will so when the next one has already started. If, moreover, the agent's subsequent action arrives in time, two commands will have arrived within the time step's 100ms time window in which case the Soccer Server executes the former and simply discards the latter.

While we stressed above that a hole *may* indicate some problem, a *clash* certainly does so as one of the commands issued by the agent will be dropped. An overview of a selection of situations in which holes and clashes occur, is provide in Fig. 3. The flipbook example from RoboCup 2022 delineated above and visualized in Fig. 2, by the way, represents a classic clash.

3.2 A Techno-Archaeological Study on Holes and Clashes

In the first years after the start of the RoboCup initiative in 1996, soccer simulation teams primarily fought with unreliable network connections between server and host machines and with delayed or even dropped packets that resulted in both, holes and clashes. As network technology became more and more reliable, around 2001/02 the focus started to shift towards computational issues and the task of guaranteeing to cease an agent's inferencing and calculations within the 100ms of the current time step.

Methodology: In the first quarter of 2024, we performed an extensive techno-archaeological study to investigate how many holes and clashes occurred during historic RoboCup events. For this, we retrieved all historical RoboCup world championships log files which are available online on `chaosscripting.net` (covering the very beginnings in 1996 and ceasing in 2017) as well as on `robocup.info` (covering competitions from 2004 till the–to that date–very recent tournament

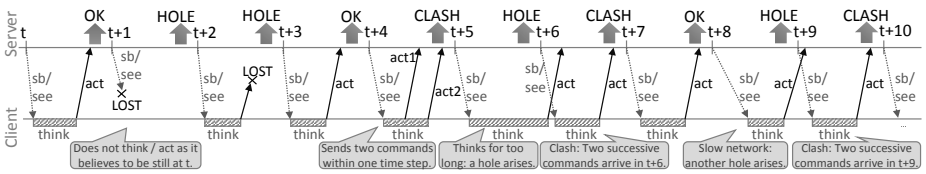

Fig. 3. The Soccer Server maintains the beat by sending "sense body" (sb) and "see" messages to clients at the beginning of a cycle. These messages are received sooner or later by an agent, depending on the network, or, in extreme cases, may get lost $(t+1)$. When received, the agent takes this as the start of the next time step, incorporates the transmitted data into its world model, starts calculating its next action and, finally, sends it to the server. The required calculation (think) should be finished fast (i.e. in less than 100ms) to ensure that the belonging act message arrives at the server within the current cycle. Holes occur when messages got lost $(t+2)$, which as pointed out in the text happens rather seldomly nowadays. Holes do also occur when an agent's command arrives too late at the server, either because of a slow network $(t+8)$, which is also unlikely nowadays, or due to too demanding computations by the agents, finished too late for the corresponding message to arrive at the server within the current cycle $(t+5)$. Clashes may occur for agent-internal reasons, for example if it (erroneously) sends multiple commands within a single cycle $(t+4$, seldomly observed in practice). If, however, a clash occurs immediately after a hole (sometimes also called a "black hole"), then most certainly something "is wrong" with the network, the timing, the computational demands, or generally with the synchronization of server and clients $(t+6$ and $t+9)$.

in 2023) [1]. In total, we focused on the almost 3400 official world championships matches that took place on-site in the previous 28 years. We omitted friendly matches as well as qualifying matches since those took place mostly on different hardware set-ups prior to the actual championships. Also, we had to skip a few of the very early years where the log file format was subject to heavy changes and was preserved only in binary format where it would have meant enormous effort to analyze and interpret the artifacts of that time with today's tools.

In the first years our study covers, holes were prevalent (e.g. the 1996 tournament saw more than 1.6 million holes in total), but their frequency of occurrence diminished over the years (e.g. from 6.600 holes per team per match on average during RoboCup 2001 in Seattle to 32 during the RoboCup 2023 in Bordeaux, cf. Fig. 4, top). Yet, these numbers are a volatile basis for further analyses, since holes are counted, too, if some player erroneously disconnects or gets trapped in an endless loop due to a programming bug and, henceforth, sends no more commands at all to the server. Therefore, all our further analyses focus exclusively on clashes occurring immediately after holes (aka black holes) as this combination hints most likely to problems in timing and synchronization of agents and the server.

Even when focusing on clashes solely, it is hard to draw general conclusions because team-internal problems and bugs might distort any statistic on clashes. We therefore conducted an analysis to identify outliers, i.e. teams whose binaries

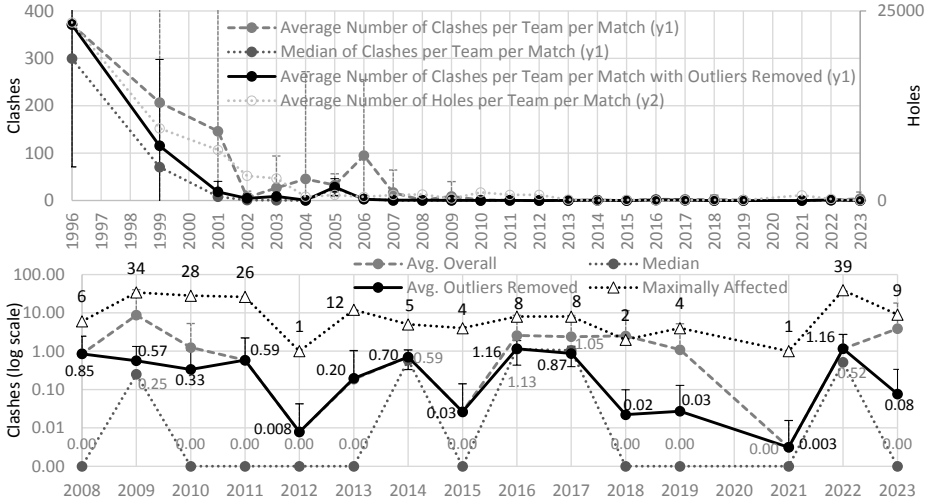

Fig. 4. Historical Review on the Occurrence of Holes and Clashes: The top chart witnesses that the average number of holes per team per match (secondary ordinate) has diminished substantially over the years. More relevantly, both charts highlight the development of clashes (black holes): The median and average number of clashes per team per match has seen a strong decline in RoboCup's first decade. However, the problem of their occurrence has never fully disappeared. Using a log scale ordinate, the bottom chart puts an emphasis on the more recent 15 years and also includes data on games that were maximally biased by clashes (after having removed outliers).

performed extraordinarily conspicuously in terms of producing holes and clashes, using the local outlier factor (LOF) algorithm [2]. Although it is not guaranteed that all teams with severe internal problems are identified this way, we empirically found that several "beknown" outliers were successfully filtered out using this approach, including Damavand'23 (heavy problems on the last tournament day), FRA-UNIted'18 (mistakenly left logging enabled in first rounds), or problems of Java-based teams like KickOffTUG'09 to name just a few.

Discussion: Essentially, when observing multiple teams with a moderate number of clashes within one season's tournament, this is a strong indication that there is or, better, was a general issue with the competition set-up. Fig. 4 (bottom) visualizes our findings: We report both, the number of clashes per team per match averaged over all teams (with and without outliers) as well as this value for the median team. Additionally, we have plotted the the maximum number of clashes of the non-outlying teams over all matches – a value that can be used to confirm or reject the conclusion that competition results were, at least in part, biased by clashes. So, for example, teams YuShan'22 and FRA-UNIted'22 had to face 4.6 and 4.2 clashes per match on average, but there were matches with as many as 39 or 32 clashes. The 2022 RoboCup tournament stands out insofar as two of the considered measures, the average number of clashes per

team per match, as well as the maximum amount of clashes in a game (outliers removed in either case), obtain an all time high in that year for the period of the past 15 years. In further analyses, which we do not discuss in detail here due to space constraints but refer to in [5], we found that a team's performance is negatively correlated with the number of clashes within that team. We also discovered that limiting the computational time available to an individual agent can easily provoke the occurrence of clashes and that Docker-based environments may be susceptible to such issues.

4 Conclusion

In this team description paper we have outlined the characteristics of the FRA-UNIted team participating in RoboCup's 2D Soccer Simulation League. After a brief presentation of the fundamental architecture of the agents, we emphasized that our recent work has focused on the agent-side implementation of the bipedal dash model, as well as on in-depth and detailed investigations of synchronization issues between agents and the server.

References

1. The RoboCup Online Archives of Soccer Simulation 2D Binaries and Log-files (Last accessed: 06/2024), http://archive.robocup.info/Soccer/Simulation/2D/, https://chaosscripting.net/files/competitions/RoboCup/
2. Breunig, M., Kriegel, H., Ng, R., Sander, J.: LOF: Identifying Density-Based Local Outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. pp. 93–104. ACM, Dallas, USA (2000)
3. Gabel, T., Breuer, S., Sommer, F., Godehardt, E.: FRA-UNIted - Team Description 2019 (2019), Supplementary material to RoboCup 2019: Robot Soccer World Cup XXIII, LNCS, Sydney, Australia
4. Gabel, T., Godehardt, B.E.E.: FRA-UNIted - Team Description 2024 (2024), Supplementary material to RoboCup 2024: Robot Soccer World Cup XXVIII, LNCS, Eindhoven, The Netherlands
5. Gabel, T., Godehardt, E.: Holes and Clashed in Simulated Soccer: When a Real-Time Simulation System Meets Compute-Hungry Agents. In: RoboCup 2024: Robot Soccer World Cup XVIII, Proceedings of the RoboCup International Symposium 2024. Springer, Eindhoven, The Netherlands (2024)
6. Noda, I.: Soccer Server: A Simulator of RoboCup. In: Proceedings of the AI Symposium 1995. pp. 29–34. Japanese Society for Artificial Intelligence (1995)
7. Riedmiller, M., Gabel, T.: On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup. In: Proceedings of the 3rd IEEE Symposium on Computational Intelligence and Games (CIG 2007). pp. 68–75. IEEE Press, Honolulu, USA (2007)
8. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement Learning for Robot Soccer. Autonomous Robots 27(1), 55–73 (2009)
9. Sutton, R., Barto, A.: Reinforcement Learning. An Introduction. MIT Press/A Bradford Book, Cambridge, USA (1998)