Research Article

Generating a dataset for learning setplays from demonstration

Marco A. C. Simões[1] · Jadson Nobre[2] · Gabriel Sousa[2] · Caroline Souza[2] · Robson M. Silva[2] · Jorge Campos[2] · Josemar R. Souza[2] · Tatiane Nogueira[3]

Received: 31 July 2020 / Accepted: 10 April 2021
Published online: 05 May 2021

Abstract
Coordination is an important requirement for most Multiagent Systems. A setplay is a particular instance of a coordinated plan for multi-robot systems in collective sports. Setplays are usually designed by robotics specialists using some existing tools, like the SPlanner, or by hand-coding. This work presents recent improvements to the Strategy Planner (SPlanner) and its corresponding FCPortugal Setplays Framework (FSF) to provide sophisticated setplays. This toolkit is useful to design strategic plans for robotic soccer teams as a particular case of Multi-Agent Systems (MASs). The new enhancements enable more realistic setplays, including, but not limited to, the definition of better pass strategies and defensive setplays. The enhanced tool is used to populate a dataset with demonstrations made by soccer experts and used in a Learning from Demonstration (LfD) approach to allow robotic soccer teams to learn new setplays. A new demonstration mode in the RoboCup Soccer Simulation 3D (SSIM3D) viewer RoboViz was also introduced to integrate this tool with SPlanner. Domain experts can use this set of tools to capture a specific scene in a game in RoboViz and use it as an initial step for a new setplay recommendation in SPlanner. The resulting dataset is organized into fuzzy clusters to be used in a reinforcement learning strategy. This paper describes the whole process.

Article Highlights

- This paper's main contribution is generating a dataset of setplays to supportlearning from demonstration in robotic soccer.
- A set of new features were added to the Strategic Planner(SPlanner) to enablethe design of more realistic setplays.
- The official RoboCup viewer (Roboviz) was integrated with SPlanner using anew *demonstration mode*.

Keywords Multi-robot systems · Strategy · Planner · RoboCup · Robotic soccer · Fuzzy clustering · RoboViz

This project is partially funded by FAPESB/IC and UNEB/PICIN. We acknowledge Vitor Manoel dos Santos for his valuable contribution during the paper review and experiments set up.

✉ Marco A. C. Simões, msimoes@uneb.br; Jadson Nobre, jadsonnobre098@gmail.com; Gabriel Sousa, gabrielmascsousa@gmail.com; Caroline Souza, carryuneb@gmail.com; Robson M. Silva, robsonms@uneb.br; Jorge Campos, jorgeapcampos@gmail.com | [1]ACSO/Uneb - Bahia State University / PGCOMP/UFBA - Federal University of Bahia, Salvador, Brazil. [2]ACSO/Uneb - Bahia State University, Salvador, Brazil. [3]PGCOMP/UFBA - Federal University of Bahia, Salvador, Brazil.

1 Introduction

Multi-Agent Systems (MASs) is a computational combination of interacting agents. In these systems, one can assume agents are both autonomous and collaborative, which means they are capable of making independent decisions and cooperating with other agents to achieve the designed goals [23].

The design of MASs is usually accomplished through predefined cooperative plans, which use different strategies such as pattern of policies [7], learning agent pairs [13] , mixed of single and coordinated agent learning process [24], game-theoretic approach [26], and multiagent planning with uncertainties [27].

The coordination among agents is critical for developing large-scale, distributed, and complex MASs. The RoboCup Simulation Leagues can be considered a suitable framework to develop new solutions for coordination in MAS. The Soccer Simulation League,[1] for instance, presents a scenario where an agent represents each robotic soccer player and a MAS represents each team. In this scenario, the agents work alongside a common objective to overcome the adversary. In the context of soccer simulations, a coordinated plan is known as a setplay. Setplays have been successfully used in Robocup Soccer Simulation Competitions [1, 10, 11, 15, 19]. The design of setplays from scratch, however, is time-consuming and hard to be accomplished. Simões and Nogueira [21] have proposed a solution to enable domain experts (both robotic and human soccer experts) to watch matches between robots' teams and report situations where they identify good moves. All these moves populate a dataset of experts' recommendations used as input of a Learning from Demonstration (LfD) engine. In order to support this approach, Simões et al. [22] proposed a new dataset schema designed to represent relevant features of a setplay. The authors used the RoboCup Soccer Simulation 3D (SSIM3D) competition rules and its associated software to validate their proposal. The 3D competition was chosen as a test-bed because it is more similar to real soccer games than the 2D competition. Humanoid robots are the base of SSIM3D rules; thus, the team's tactic and individual player skills become relevant to perform well in this competition.

Aiming at validating the aforementioned LfD approach, a case study with Bahia Robotics Team (BahiaRT)[2] was carried out. In this case study, the Strategy Planner (SPlanner) [6][3] and the FCPortugal Setplays Framework (FSF) [16] were used to populate the setplay dataset. The existing version of SPlanner presents several drawbacks to implementing the LfD approach. This version does not support opponent teams, which means that only one team is allowed in the field. This limitation does not let setplays designers exploit opponents' positions to describe bad situations and recommend good behaviors. The version also lacks defensive setplays actions. Moreover, there is no option for passes when the SPlanner's user wants to delegate the receiver player's decision to the team.

One of the significant challenges of this work is to translate soccer expert's common-sense knowledge into a set of recommendations for the LfD engine, which will become planning actions of setplays of a MAS. Common-sense knowledge is the set of all techniques and wisdom that experts acquire during their lifetime. This knowledge is not easily formalized or stated in the form of generic rules. Aiming at converting this knowledge into a recommendation, we provide domain experts with pre-recorded videos or simulations of a MAS in action and ask them to stop at any point where they think the robots are not performing well. The expert then can write down, using a domain-specific schema, the best recommendation to fix the wrong move. The idea is to capture real situations and set recommendations of coordinated plans to increase the MAS performance.

One can use SPlanner and FSF to design and run a setplay from scratch, starting with an empty field. In this case, the designer should imagine the entire game situation to start the setplay. This work proposes that domain experts can watch the MAS in action and then capture a specific situation where he thinks the robots are not doing well. The initial scene of this situation must be captured and launched in the SPlanner. Thus, the setplay designer can start from a predefined game situation to create a new setplay. A new demonstration mode was added to the official SSIM3D viewer RoboViz[4] to support the integration with the SPlanner. This new demonstration mode integrates the RoboViz and the SPlanner and makes a set of new features available to users.

Some of the enhancements to turn the SPlanner eligible to support this work were already presented [20]. This paper updates these enhancements, filling all gaps. Using the integration of Roboviz and the new version of SPlanner, users can now create recommendations of setplays to populate a consistent dataset. We have updated the dataset schema presented in our previous work [22] and performed some assessment to validate its organization in fuzzy clusters. These results confirm that the dataset we can generate using all the methods described in this paper

[1] See https://ssim.robocup.org.

[2] http://www.acso.uneb.br/bahiart.

[3] Available at https://sourceforge.net/projects/fcportugalsplanner/.

[4] Available at: https://github.com/magmaOffenburg/RoboViz.

is ready to be used as input for a reinforcement learning strategy. This strategy is the final step in our project to transfer knowledge from domain experts to robots using demonstrations.

This paper's remainder is structured as follows: Sect. 2 discusses some related work concerning techniques and tools for setplay learning. Section 3 presents the improvements added to SPlanner to support setplays' development based on LfD approach. Section 4 details new features added to RoboViz. Section 5 presents a dataset schema used to organize the setplays' dataset and make it adequate to use in a reinforcement learning approach. Section 7 draws conclusions and indicate future works.

2 Related work

MAS, in general, and MAS planning, in particular, has been a very active area of study in the artificial intelligence community. Many researchers have investigated planning considering a large number of agents. A scalable solution to large teams (1024 agents), for instance, considers the team geometric pattern instead of individual agent positions for multiagent learning [7]. A neural network model named HyperNEAT was used to generalize the agents' roles in the MAS from the system's agents' positioning. The authors argue that their contributions are conceptual. The model was not validated in real applications or well-known challenges in the scientific community.

Another work [26] describes the use of Bayesian networks to learn opportunistic criminals' behavior in an urban area. The idea is to plan the schedule of patrol units based on the criminals' behaviors. This work was further extended [25] to use the *Expectation Maximization*(EM) algorithm alongside Bayesian networks to enhance the learning of opponent agents behavior (e.g., opportunistic criminals in urban areas). Opponents' behavior learning is also explored using Markov chain models with Monte Carlo methods to support MAS planning [17]. MAS planning is alternated with opponents' behavior learning to feed the plans generated. None of these approaches learn new cooperative multiagent plans. They use learned information about opponent behaviors to support the classical multiagent planning process.

A model for concurrent planning in a MAS was presented using two learning approaches: Monte Carlo and LfD [24]. The authors validated this work in a static manipulator robots domain, where the robots cooperate in a MAS to assemble an object. The nature of data in this environment is different from the data in mobile robots of a Multi-Robot System (MRS) in a stochastic, real-time, partially observable environment (e.g., robotic soccer). This work does not treat semantic equivalence or any similar issue, which means that it does not identify when domain experts' recommendations are slightly different but have similar semantic meanings.

The interdependence of agents' behavior in a MAS was explored using the Q-Learning algorithm and distributed Bayesian networks to model supply chains' planning in a global product sales market [28]. Another approach [14] investigates fault (and their causes) detection in MAS plans. The fault detection is associated with agents' actions interdependence. Bayesian inference was used to diagnose the MAS faults and their causes. On the one hand, none of these solutions provides new coordinated plans. On the other hand, both work with learning or reasoning about behaviors interdependence, which can help deal with abort conditions in a setplay.

Synergy Graphs were presented for real-time coordination between agents in a MAS [13]. The work used the well-known multi-armed bandit problem for validation. The authors did not mention if the learning process is restricted to the coordination between agents or if the process learns full coordinated plans with all their steps and agents' behaviors. In any case, no demonstrations from domain experts were used.

A MAS with a set of moving agents must have a set of functions, information, strategy models, among other aspects [21]. Since there is a substantial number of components, the MAS also needs to consider coordination to work cooperatively. Cooperation relies on many communication protocols so that the agents can agree with each other and accomplish their objectives. For instance, in a soccer game, the team may decide whether or not to be more aggressive (i.e., changing the behavior of selected players) in the case in which it is losing the match [22].

In robotic soccer, the complexity of creating setplays and the fact that every team has its design methods led to creating a standard language for setplay developments. A framework to produce setplays named SPlanner was proposed by Cravo et al. [6]. SPlanner was designed to make setplays creation faster, easier, and more intuitive. Thus, even those with no knowledge about robotic soccer can develop their setplays.

SPlanner [6] is composed of a syntactic analyzer, an interpreter, and a real-time selection and execution software. This framework also has a pattern of predefined general-purpose behaviors. In this way, teams' developers can easily map the behavior language to their teams' algorithms, making the design and usage of setplays easier. The authors validated this tool in RoboCup competitions showing good results [6], both in the 2D Soccer Simulation and Middle-Size soccer. A recent work [18] has also extended the validation of SPlanner to the SSIM3D competitions, adapting BahiaRT to use FSF. To

Fig. 1 Adversary players board

validate this extension, the authors have developed some simple setplays using SPlanner.

Machine learning has already been used in setplays development or optimization context. Automatic analysis of match logs was used to generalize a setplay in a simulated robotic soccer environment from a sequence of successful events [1]. A sequence of behaviors derived from coordinated positioning is formalized as a plan and incorporated into the team's setplay library for future use. The knowledge used emerges from the agents' natural interaction. These agents do not use any domain expert knowledge for coordination.

SPlanner and FSF were also used to support experiments with multiagents. In [10], multi-agent Q-Learning was used to learn a transition function for multi-flow setplays. In multi-flow setplays, each state can lead to more than one following state, depending on the transition conditions. In that work, these transition conditions were generalized using reinforcement learning. However, the proposal does not present how agents can learn a complete setplay.

Simões and Nogueira [21] proposed a complete framework to learn new setplays from a set of demonstrations created by domain experts. Simões et al. [22] have also presented a dataset schema to represent these demonstrations in a LfD engine. The authors defined that two setplays are semantic equivalent if they represent the same gameplay and the same teams' goals in a specific situation. Their approach organizes the dataset into fuzzy clusters, grouping semantic equivalent setplays. The next section describes the enhancements made in SPlanner to support experiments using the LfD engine with the proposed dataset schema for setplays.

3 Enhancements in SPlanner for better passing and defense strategies

The main focus of this paper is to make the SPlanner useful to populate datasets [22] based on robotic soccer domain experts' demonstrations. We also intend to make SPlanner more adequate for using in the SSIM3D competition and fill some gaps that limit its use in robotic soccer matches. To achieve these goals, changes in SPlanner are developed as follows.

- Opponent Team: some defensive actions were dependent on an adversary player. Thus, the possibility of placing enemy agents on the setplays was added to make these actions viable, as seen in Fig. 1.

  A full team of 11 opponents using blue shirts was added along with a board for them on the side of the field. Each opponent has three possible behaviors:

  - Run—makes the player move to a targeted position. This action offers to the setplays designer the option of estimating the opponents' movements from one setplay step to the following one. This action can be used both in offensive and defensive setplays.
  - Kick—kicks the ball to another opponent player. The purpose is to let the user estimate passes done by the adversary team. This behavior is available only for the opponent who owns the ball in defensive setplays.
  - Shoot—the setplay designer can use this behavior to estimate a situation where an opponent can shoot to the goal. This action is available only for the adversary player who owns the ball in a defensive setplay.

  Fig. 2 shows the opponents' behaviors menu.
- Defensive Setplays: there was a defensive setplay option on the previous version of SPlanner [6], but the available version of this tool does not implement it. Defensive setplays are as crucial to the team's strategy as offensive setplays are. We have changed SPlanner to effectively implement the defensive setplays to all

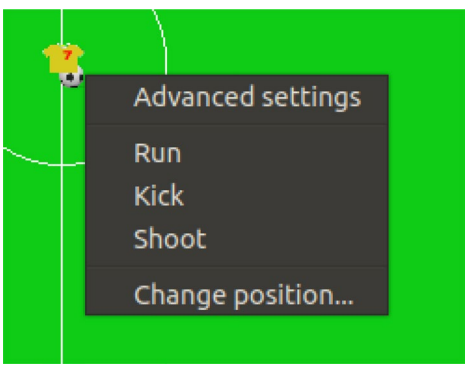


Fig. 2 Opponent behaviors' menu

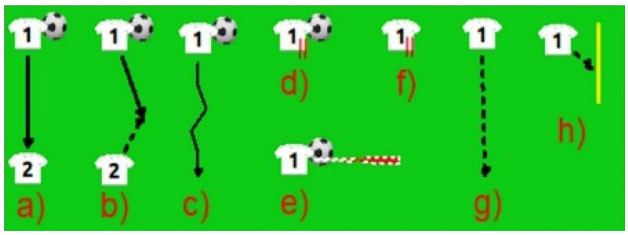


Fig. 3 SPlanner's base version actions: **a** pass, **b** pass forward to <player number>, **c** dribble, **d** hold, **e** shoot, **f** wait, **g** run, **h** go to offside line

situations where an offensive setplay was available. This option has highly amplified the team's capacity to perform setplays at any time in a game. One important feature to support defensive setplays is the presence of opponent players, since defensive behaviors can use adversaries as reference. For example, one teammate can perform a defensive marking action during a setplay to follow an opponent in the field and prevent him from receiving a pass.

– New Behaviors: SPlanner came along with eight actions that could be executed by each player in its base version, as can be seen in Fig. 3 [6]. These actions are:

a. pass: perform a pass to a specific teammate;
b. pass forward to <*player number*>: pass to an advanced point in the field where the teammate chosen to receive the pass is supposed to move to intercept the ball;
c. dribble: carry the ball on avoiding opponents;
d. hold: standstill while keeping the ball possession;
e. shoot: shoot to the opponents' goal;
f. wait: standstill in the same place;
g. run: move to a specified position;
h. go to offside line: move to a position just behind the offside line.

Besides these actions, new ones were added to make the toolkit more suitable for the 3D league and for populating the LfD engine dataset. Fig. 4 shows the modified behaviors menu for offensive setplays. There are two versions of these menus: one for players without ball possession (*A*) and another for players with ball possession (*B*). Each of the new implemented actions is described as follows:

– Run - Straight:
This action is a new walking/running method. Many teams have at least two types of walking: one is slower and more reliable, with little to no falls, and the other is faster, closer to a running skill, but with less balance. *Run–Straight* is a behavior created to map the fastest walking/running behavior present in the team. This behavior, in most cases, prioritizes velocity to balance, and usually, the player is not aware of collision avoidance.
– Run - Path planned:
We graphically changed the name of this behavior from "run" to "run - path planned" so that the

Fig. 4 New Behaviors for offensive setplays: **a** behaviors menu for players without ball possession; **b** behaviors menu for players with ball possession

user can notice the difference from this walking/running action to the *Run - Straight* behavior. In the *Run—Path planned*, the player prioritizes balance to velocity avoiding collisions.

– Offensive Marker:
  BahiaRT's set of behaviors inspired this action. The main goal is marking enemy agents that impose the risk of taking the ball possession from one of our teammates. The player moves to a strategic position between the enemy and the player with the ball, blocking the adversary while keeping a certain distance so it won't affect the freedom of the player with the ball to kick, do a pass, or any other action it might take.
– Pass To Best Player:
  Just like the offensive marker, this action also comes from the BahiaRT. It uses a selection algorithm to choose the ally in the best conditions to receive a pass. When used correctly, this action can result in excellent plays in a game. When the setplay designer uses this behavior, he should create multiple transitions for the current step where this behavior is used. Each transition takes the setplay to a next step considering a different teammate chosen to receive the pass.
– Intercept: this behavior is not exhibited in any behaviors' menu. It is an implicit behavior related to the *Pass forward to <player number>* action. When the user assigns the *Pass forward to <player number>* and selects a teammate to receive the ball, this receiver is implicitly assigned to the *Intercept* action. The receiver is supposed to move to the region where the player who owns the ball passes the ball to and then try to intercept it. In this version, the *Intercept* receives the region to where the ball is passed as a parameter. This modification makes it easier for the teams to implement this behavior when extending the FSF.

Some new behaviors were also designed for defensive situations. Fig. 5 shows the menu for defensive situations. These new actions can be described as follows:

– Defensive Marker:
  This action is concerned with marking opponents in defensive situations. It can be split into two types: the active marker when the player marks the adversary with the ball possession trying to regain possession of the ball; and the passive marker when the player marks a potential enemy receiver for a pass executed by the adversary, thus disturbing any opponent plays. In the SPlanner, only a general defensive marker behavior is used. When a team extends the FSF, it can provide the two specializations (active and passive) of the defensive marker behavior.
– Become Owner:
  This action switches the owner of the ball to the player who executes this behavior. It was designed for use in defensive setplays. The purpose of this behavior is to make a teammate take the ball possession of an opponent. In general, this is the last action used in a defensive setplay, once the main goal of a defensive strategy is to take the ball possession back to our team.

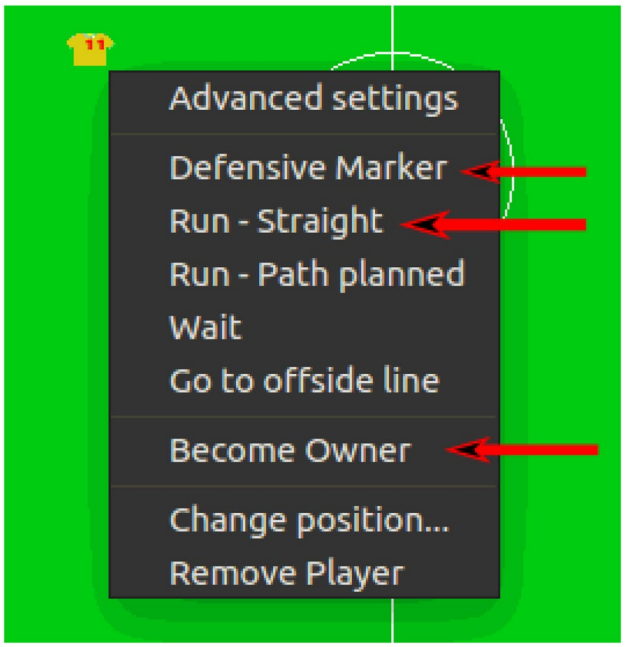


Fig. 5 New Behaviors for defensive setplays

– Step Abort Conditions:
  In the base version, this field was initially called "Step Times" [6]. But we realized the need to add a "Step Abort Conditions" field. Since those times where also abort conditions, we also decided to reuse the field changing the name and adding what we needed. In a setplay, there are conditions for it to begin executing (start conditions), conditions that assure its continuity (transition conditions), and conditions that once met are supposed to abort the execution of the setplay (abort conditions). SPlanner had general abort conditions, for instance, when the enemy steals the ball. But, there were no specific abort conditions for each step of the setplay, and those were necessary since there are situations during a specific step where the desirable conditions are not met, and the setplay should

Fig. 6 New "Step Abort Conditions" field with the "passFailed" condition

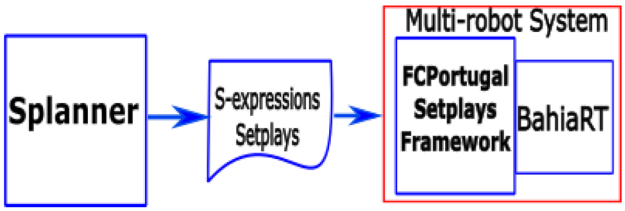


Fig. 7 SPlanner generates a s-expression file containing a setplay to be interpreted by FSF and executed by BahiaRT

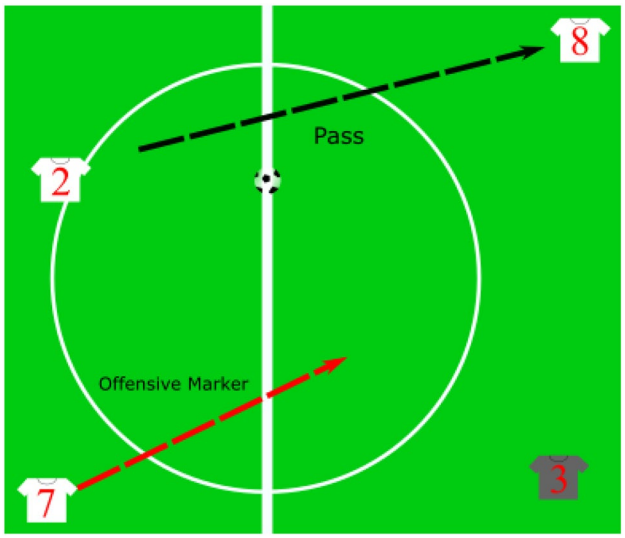


Fig. 9 Setplay using the new behavior Offensive Marker. The player number 7 uses the Offensive Marker to block the opponent and let the player number 2 to perform a pass

be terminated, as an example, a pass that was poorly executed. To fix that, we added to the "Step Abort Conditions" field the "passFailed" option, which terminates the setplay in case of a bad pass (Fig. 6).

All these changes in SPlanner must be mirrored in the FSF as changes in the *libSetplay*[5]. When the design of a setplay is finished in SPlanner, the user can export it to a text file using a specific language defined by the FSF [6]. This language is based on an s-expression syntax. Figure 7 shows the relations between SPlanner, FSF and BahiaRT.

The language used to represent the setplay in a file was extended to support new behaviors described in this section. The syntax for these new and modified actions is exhibited in Fig. 8. This syntax is the way the actions are generated and read by the FSF. This extension in the *libSetplay* turns all the enhancements presented in this work available for any robotic team that can extend the FSF.

The following section presents some examples of use that take advantage of the new features added to SPlanner.

3.1 Examples of use and discussion

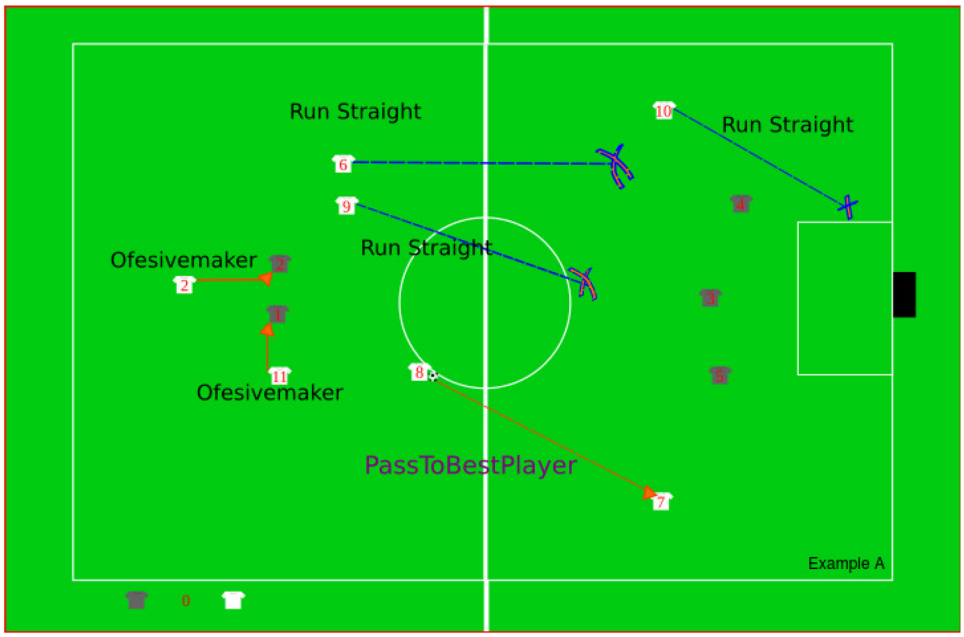
All enhancements incorporated in SPlanner come along with tutorials describing possible situations where they can be used. We selected some of these examples to discuss such enhancements.

The first example shows the usefulness of Offensive Marker behavior (Fig. 9). The player number 2 tries to perform a pass, but the opponent number 6 can try to block this pass. So we have used the Offensive Marker behavior

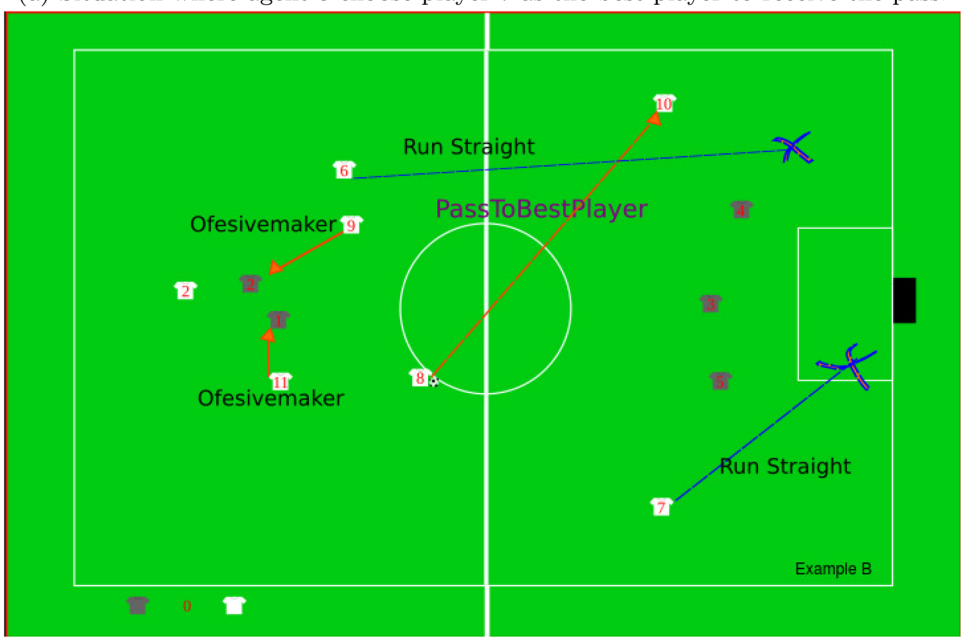
Fig. 8 New actions syntax

```
defMark:(defMark : destPlayer <PLAYER-REFERENCE>: relTo <TYPE> )

passtobestplayer:(pbplayer :player <PLAYER-REFERENCE> :type <TYPE>)

run Straight: (mov :region <REGION>)

intercept: (intercept :region <REGION>)

ofensive marker: (ofeMark : destPlayer <PLAYER-REFERENCE>: relTo <TYPE> )
```

[5] *libSetplay* is an open-source library where the FSF is implemented



(a) Situation where agent 8 choose player 7 as the best player to receive the pass.



(b) Situation where agent 8 choose player 10 as the best player to receive the pass.

Fig. 10 Examples of use of the behaviour *PassToBestPlayer*

in the teammate number 7 to block the movement of the opponent and let the player number 2 free to execute the pass.

Humanoid robots can not perform passes so quickly as wheeled robots. A humanoid needs to position itself and then control a set of joints to perform the kicking movement. Opponents can reach the ball before our player can complete the kicking movement most of the times. For this reason, the offensive marker is a vital behavior to enable the success of ball passing. The pass is an essential movement in most setplays. The addition of the offensive marker behavior increases the usefulness of the FSF and SPlanner to the humanoid robots' soccer domain.

The second example demonstrates the use of the *Pass to Best Player*, as seen in Fig. 10. The image shows a common situation where player number 8 is with the ball and has to choose whether he passes it to player number 7 or player number 10. The SPlanner has a resource called multi-flow, where you can split the setplay course of actions depending on its flux. The addition of the *Pass To*

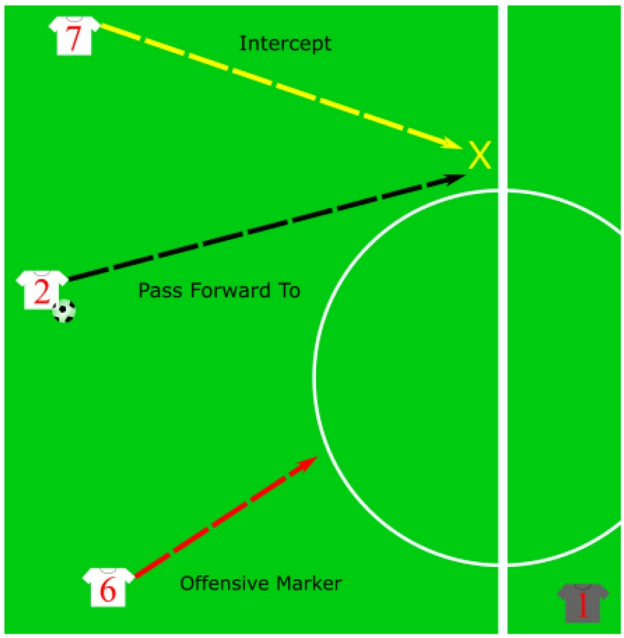


Fig. 11 Example of a setplay using the existing *pass forward to* behavior with the new actions *intercept* and *offensive marker*

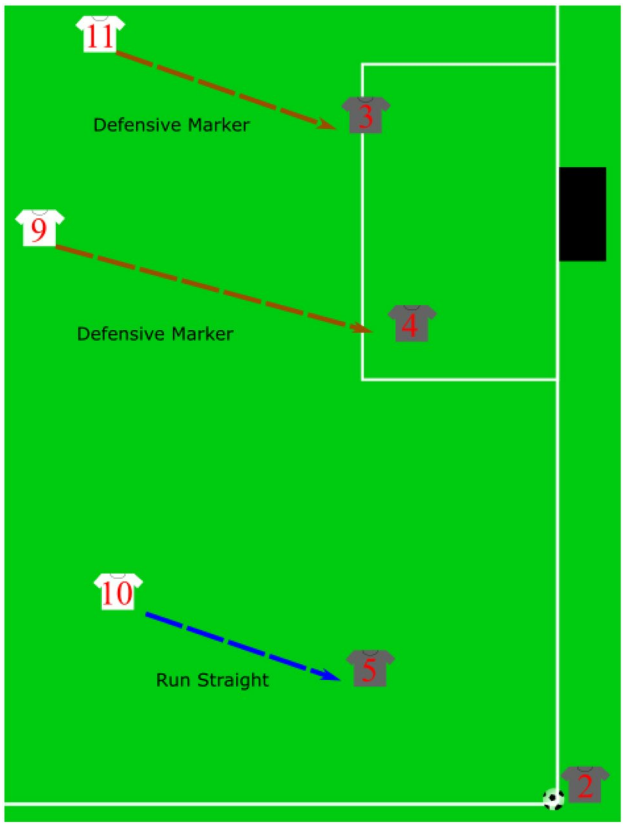


Fig. 12 Example of setplay using the *defensive marker* behavior and the *run - straight*

Best Player takes a great advantage of the multi-flow. Interesting usage situations arises when the setplay designer wants to delegate to MAS the task of evaluating the targets for a pass, and decide a different flow for the setplay depending on this decision. The designer can plan two or more different flows and the MAS takes the decision of which one the agents will follow according to the chosen player to receive the pass.

In example Fig. 10a, we can observe that player number 8 picked player number 7 as his pass target, thus, this player will continue the setplay flux as the new ball owner. Fig. 10b shows a situation where player number 10 was chosen instead of number 7. The teams' algorithm is responsible for analyzing which teammate is the best one to receive the pass. When the designer picks a teammate by himself, he assumes that the team will be able to execute that specific action exactly the way he planned, but that is not always true. As mentioned before, the *Pass to Best Player* allows the setplay designer to delegate to the MAS the decision of which teammate to receive a pass. Delegating this choice to the team's AI makes this action far more reliable, as it increases its chance of success.

On the third setplay (Fig. 11), there is another situation where the objective is to execute a pass. The player number 2 is going to execute a *pass forward to* player number 7, who is going to move to the region informed by player number 2 using the new *intercept* action. Since the enemy is showing a potential threat, the player number 6 executes an offensive marking to protect player number 2 pass attempt. In this example, the main modification demonstrated is the use of the new *intercept* action. In this version, the setplay will define that the player who will try to intercept the ball will move to a specific field region where is the target of the ball passing. The player can start moving towards the target region earlier and the chances of success in the ball interception are increased.

The last example (Fig. 12) shows a defensive corner setplay. In this example, player number 10 approaches the region where the kick is going to happen in order to disturb the enemy player supposed to receive the pass. Player 10 uses the *run - straight* since there are no obstacles between him and the target region. The players number 9 and 11 execute the *passive Defensive Marker* to prevent other enemy agents from receiving a pass, and while doing so, they also try to regain the ball possession from the enemy team.

In this example, the *run - straight* and *defensive marker* are demonstrated. Many teams are developing or trying to develop faster running skills for their humanoid robots. The goal is to present a *sprint* skill to move a robot very fast from one region of the field to others. This kind of movement does not care about collision avoidance. A big problem for the teams when using a *sprint* is deciding when it

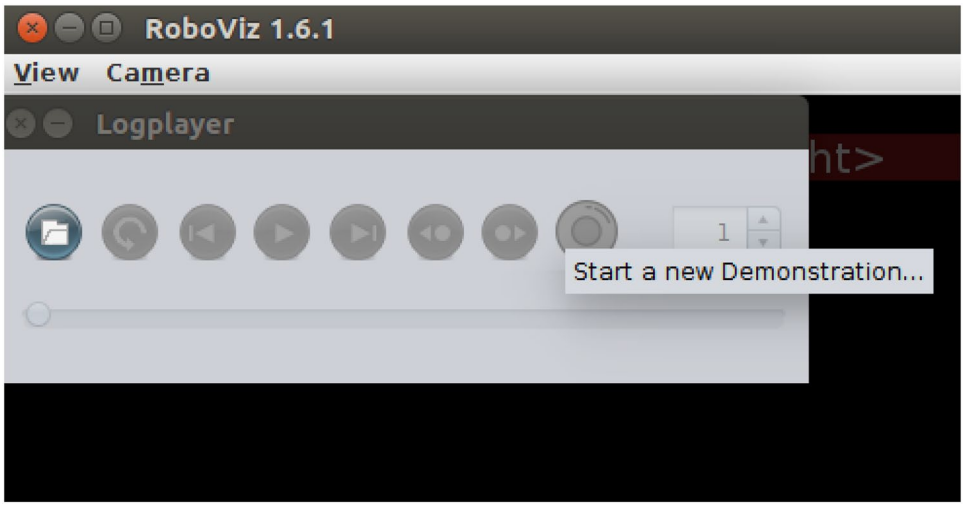


Fig. 13 RoboViz initial screen in the demonstration mode. The *Start demonstration* button can be used during log play to create a new demonstration

is useful to use this skill. When we added the *run - straight* to the SPlanner, the decision about when a robot should use a *sprint* skill can be delegated to the setplay designer. This delegation is important because the strategic choice of risky behaviors is preferable to an algorithmic decision. Reactive algorithms can not estimate all situations and the consequences of using a *sprint* behavior. The previous version of SPlanner lacked a behavior to mark opponents without ball possession. Potential receivers can not be free to receive a pass and shoot to our goal. So, the *defensive marker* is an essential feature to the setplays designer to create defensive strategies to avoid opponents' pass to be well succeeded and regain the ball possession.

4 Integrating RoboViz and SPlanner

The RoboViz is the official SSIM3D competition viewer. RoboViz is used in two operational modes: gaming and log modes. The gaming mode is used during real-time games. It is the mode use during competitions. The RoboViz connects to a simulator in the network and receives all log information while the match is processed by the simulator. RoboViz renders the graphical scene corresponding to the current game instant and displays it on a monitor. In the log mode, no simulator or online connections are necessary. The RoboViz opens a log file previously generated by the simulator and renders the match recorded in this log file. From the user point of view, it is like a replay of an old match. The log mode is very useful to developers and researchers when they are trying to understand the limitations of their teams to define strategies and enhance their performance. They can watch matches played during competitions or previous test in their labs and detect the weakness of their teams. However, when they identify any point to enhance, they need to hand code the conditions that represents that specific scene in their development tools. There is no way to automatically extract that game situation to other tools.

In this work, we develop a new demonstration mode in RoboViz. This mode can be used when launching RoboViz using the *–demoMode* flag. The RoboViz launches in an interface similar to the log mode but containing an extra *Start demonstration* button as shown in Fig. 13. the button is disable when the application launches. When a log file is open, the button is enabled. The RoboViz starts playing a log as soon as the file is open by the user. The remaining buttons in the Logplayer window are similar to those in an usual video player, like play/pause, rewind, fast forward, etc.

During the log play, when the user watches a situation where he wants to start a new demonstration, he clicks the new button and provide some information as shown in Fig. 14. The user should provide some initial information on this screen. He chooses one team for whom he is providing this new demonstration. Then, he chooses the type of the setplay and a play mode. The type can be offensive or defensive. When the offensive type is selected, the tool assumes that the chosen team has ball possession. Otherwise, it considers that the opponent team owns the ball. The list of play modes is context-sensitive. Only the play modes that were observed in the current log up to the current instant will be shown. If, for instance, there was no goal kick in the current match so far, the list will now show goal kick as an option. The user should choose the play mode corresponding to the point he wants the log to be re-positioned to. If you choose *BahiaRT Kick-in*, for instance, the log will reverse to the last kick-in favorable to BahiaRT.

When the user fills all the information, he can press the *Play* button to re-position the log. This process may be slower because the input for RoboViz is not a video

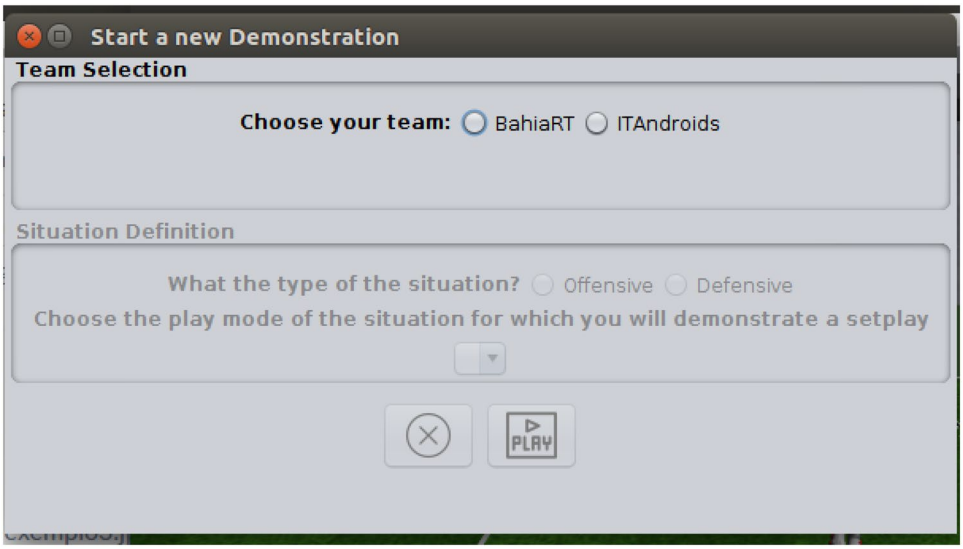


Fig. 14 Screen launched when a new demonstration is started. The user should inform for which he want to create a recommendation, the type of setplay (offensive or defensive) and the play mode where he wants the log to be re-positioned

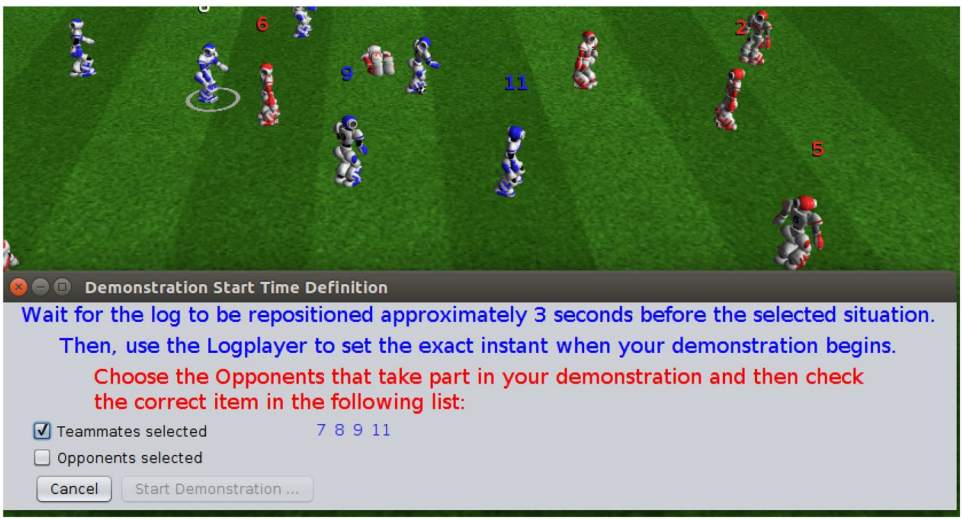


Fig. 15 After log re-positioning, the teammates are chosen. The user just click on each player and its number is inserted in a list on the site of the option *Teammates selected*. The checkbox must be selected when all teammates where chosen

streaming. The RoboViz reads a log generated by the simulator. So whenever a log re-positioning is necessary, it must build again the sequence of scenes necessary to reach the target point in the log. A new window with instructions to wait the log to reverse to the target position and to press the *Pause* button when the log is in the scene where the user wants to start the demonstration. The *forward/backward* button may be used to fine tune the exact position in the log. The Fig. 15 shows the demonstration start time definition window. When the log is re-positioned, the user can choose the players form the team he has chosen in the previous screen that will take part of the setplay. We call these players as teammates. The user just click on each player and the number of each selected robot is added to a list alongside the option *Teammates selected*. When all players are selected, the user must check the box on the left of this option.

Then, the user must select the players from the opponent team that he wants to include in the setplay and check the *Opponnets selected* option. The user pushes the *Start demonstration* button and a demo file is exported to the disk. The SPlanner is automatically launched to import this file. Figures 16 and 17 show an example of demonstration scene in RoboViz, just before exporting, and in SPlanner just after importing.

The blue and red numbers in the window on the right top corner indicate the players chosen to take part of the demonstration. Notice these players are loaded in the same relative position in the field in the SPlanner (Fig. 17).

When SPlanner import the demo file, the players considered *Teammates* will be represented by white t-shirts, and the *Opponnets* use blue t-shirts. From this point on, the user will design his setplay using regular SPlanner features. When he finishes, he exports the setplays to a

Fig. 16 Demonstration ready to be exported. Players 7, 8, 9 and 11 from blue team and players 1, 2, 3, 4, 5, 6 and 11 from red team will take part of the setplay

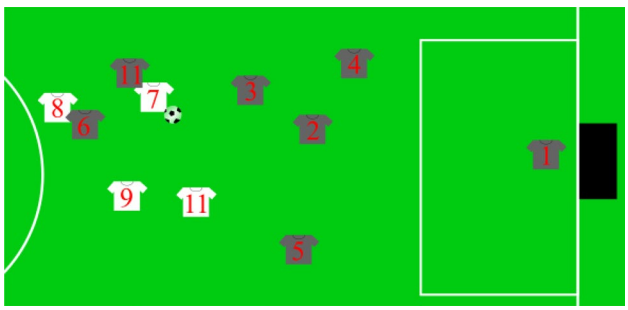


Fig. 17 Demonstration imported into SPlanner. The white players are the Teammates and blue players are the Opponents

file and closes SPlanner. The RoboViz will continue the log playing in the same point where the demonstration was started. Thus, a user can generate several demonstrations while he watches one unique match. the demonstrations are independent of each other.

The file exported by RoboViz uses an S-expression syntax that is easy to be parse by any tool. Thus, this file can be used by other tools besides SPlanner. The file is very simple as can be seen in the Fig. 18.

The file contains very basic information. Two lists of players - teammates and opponents—are in the begin of the file. Following them, there are the play mode, the leader player and the ball holder. The play mode is the one chosen by the user when started the demonstration. The leader player is always a teammate. In offensive setplays, the leader and ball holder is the same player. In defensive setplays, the leader is the teammate who is closer to the ball. The ball holder is the player who owns the ball possession. He can be a teammate (in offensive setplays) or a opponent (in defensive setplays).

The modified tools presented in both this and previous sections make it possible for a user to generate demonstrations of setplays starting from real scenes in games' logs. In next section, the overall process for generating the dataset is presented. The schema to organize the dataset is also described.

5 Organizing the dataset

Since the required tools to generate a setplay demonstration are available, a process to create this dataset must be defined. The full process is illustrated in Fig. 19.

An expert can use RoboViz to watch games logs and choose some situation to start setplay demonstration. The RoboViz exports a demonstration file and launches SPlanner. The demonstration file is loaded when SPlanner is initiated. The expert can complete the setplay recommendation and export it to a set of setplays. The expert will continue watch the game from the point where he started his first demonstration and this process will repeat while the expert wants to provide new recommendations.

Fig. 18 The demonstration file exported by RoboViz uses a simple S-expression syntax

```
(demo :name BahiaRT-400.525
        :players
                (list
                        (at (playerRole :roleName Player7) (pt :x 6.890516 :y -1.2782754))
                        (at (playerRole :roleName Player8) (pt :x 4.9517903 :y -1.0762119))
                        (at (playerRole :roleName Player9) (pt :x 6.243064 :y 0.3161399))
                        (at (playerRole :roleName Player11) (pt :x 7.497106 :y 0.45839706))
                )
        :playersOpponents
                (list
                        (at (playerRole :roleName PlayerOpponent1) (pt :x 14.6746235 :y -0.2478271))
                        (at (playerRole :roleName PlayerOpponent2) (pt :x 9.516699 :y -0.60560226))
                        (at (playerRole :roleName PlayerOpponent3) (pt :x 8.434658 :y -1.3939612))
                        (at (playerRole :roleName PlayerOpponent4) (pt :x 10.192728 :y -1.9074758))
                        (at (playerRole :roleName PlayerOpponent5) (pt :x 9.348128 :y 1.4145325))
                        (at (playerRole :roleName PlayerOpponent6) (pt :x 5.5126295 :y -0.67984545))
                        (at (playerRole :roleName PlayerOpponent11) (pt :x 6.4658666 :y -1.640686))
                )
        :condition (playm PlayOn)
        :leadPlayer (playerRole :roleName Player7)
        :ballHolder (playerRole :roleName Player7)
)
```

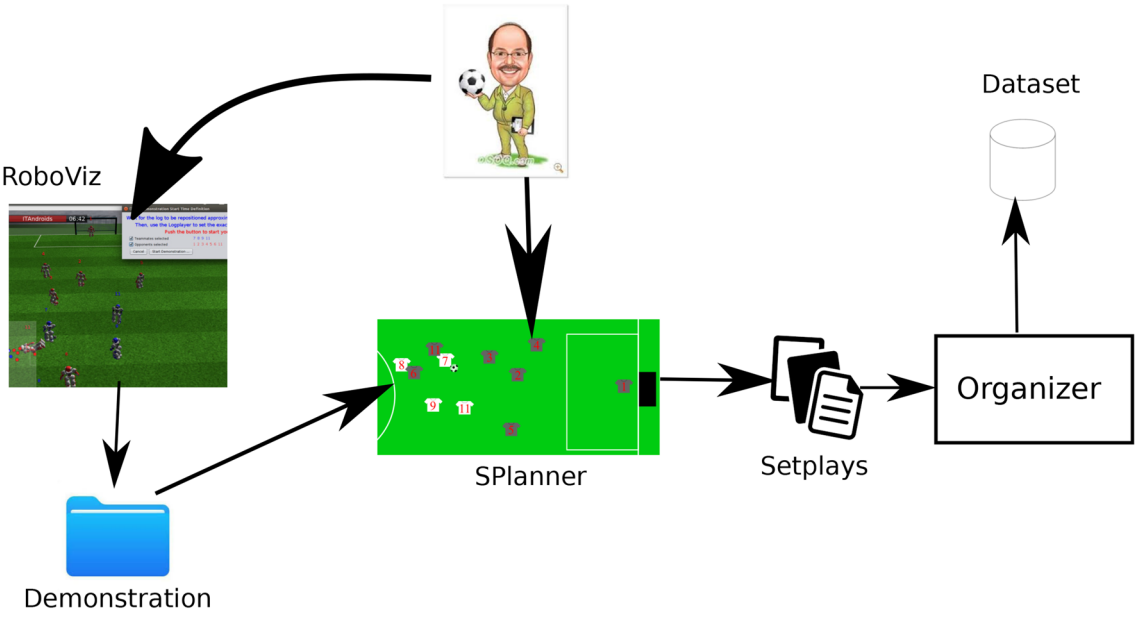


Fig. 19 The full process of generating a dataset for learning setplays from demonstration

This project uses a crowd-sourcing strategy. So the tools will be available for several people from anywhere provide recommended setplays. Thus, it is expected to get large set of setplays. These setplays must be organized into a dataset adequate for using in a reinforcement learning strategy. In a previous work, a dataset schema was presented [22]. The organizer uses this schema to generate the dataset ready for training the MAS to use the recommendations. One of the main issues solved by the new schema is semantic equivalence.

Definition 1 (*Semantic equivalence*) Two setplays σ_p and σ_q are considered semantic equivalents if they represent the same play in the domain abstract knowledge level [22].

By *domain abstract knowledge*, we mean the knowledge that specialists say they have acquired based on their experience. It is a kind of *common sense* between the majority of domain experts.

Thus, setplays considered semantic equivalent are grouped under a cluster structure. However, the concept of semantic equivalence is not precise. A setplay may present features to be part of more than one cluster in the structure. For this reason, the organizer uses a fuzzy strategy. The Fuzzy C-Means (FCM) algorithm was adapted to deal with non-scalar features and group the setplays using an hierarchical structure. Some important information from setplays (e.g. conditions) are not scalar information. In this proposal they were codified as binary trees. This required an adaption in the conventional FCM to deal with these kind of information

Other issue is that a setplay is not a linear object. For instance, it contains a list of steps where each step is a complex object with a list of features. The strategy was to define the cluster in two levels. In the first level, only the information regarding setplays identification were considered. At this point an initial set of clusters were generated. Thus, a second clustering level is executed inside each cluster generated in the first level, generating sub-clusters. In this second level, only information regarding the steps of each setplay is considered.

Table 1 lists the features necessary to represent a setplay in the dataset. The last feature(stepsList) in this table is a list of complex structures composed of several features. A secondary dataset schema described in Table 2 details the stepsList.

We have changed the feature *nextStep* since our previous work [22] to support multiflow setplays. In the updated version we represent the *nextStep* as a list of integers, while in the previous version it was a single integer. The addition of the new behavior *passToBestPlayer* (see Sect. 3) requires a full support to multiflow setplays.

Figure 20 illustrates a schematic view of the proposed hierarchical dataset schema. The schema is divided into two levels. At the first level, there are features that identify the setplays. Each instance at this level represents a different setplay recommended by demonstrators. The second level describes the steps within each setplay. The organizer reads the S-expression files and transforms them into the structure illustrated in Fig. 20.

The detailed description and validation of this clustering strategy was presented earlier [22]. In the next section, we

Table 1 Dataset features representing setplays extracted from SPlanner

Feature	Description	Data type
ourPlayersNumber	Number of BahiaRT players participating in the setplay	Integer
theirPlayersNumber	Number of opponent players participating in the setplay	Integer
abortCondition	Boolean expression representing the condition to abort the setplay	Binary tree representing the parsed Boolean expression
Steps	Total number of steps composing the setplay	Integer
stepsList	List of steps composing the setplay	Vector of Step. Each step is a composed structure presented in Table 2

Table 2 Features that composes one step

Feature	Description	Data type
ourPlayersInStep	Number of BahiaRT players participating in this step. This number must be smaller than or equal to ourPlayersNumber	Integer
theirPlayersInStep	Number of opponent participating in this step. This number must be smaller than or equal to theirPlayersNumber	Integer
waitTime	Minimum time to wait before a transition from a current step to another can be conducted	Double
abortTime	Maximum time to finish the current step	Double
ourPlayersList	List of BahiaRT players participating in this step	Vector of Player P_B. Each Player $p_B \in P_B$ is represented by a Cartesian coordinates pair (x_{p_B}, y_{p_B})
theirPlayersList	List of opponent players participating in this step	Vector of Player P_O. Each Player $p_O \in P_O$ is represented by a Cartesian coordinates pair (x_{p_O}, y_{p_O})
nextStep	Identification of the possible next steps after a transition condition is met	list of Integer
condition	Boolean expression determining the transition condition from the previous to the current step	Binary tree representing the parsed Boolean expression
behaviorsList	List of behaviors executed by each player $p_B \in P_B$	Vector of strings

describe the new assessment executed with the clustering strategy when applied to the new dataset generated with the new integrated tools. The organizer generates a dataset built in a two-level fuzzy clustering structure. This dataset will be used for training the MAS to learn a control policy to choose the correct cluster for each situation. When the policy is learned, the agents will evaluate each simulation episode to decide if they will launch a setplay from one of the clusters or continue to act in a reactive mode. If a cluster is chosen the agent will use the current Case-Based Reasoning (CBR) algorithm present in the FSF to choose one of the setplays inside the selected cluster.

6 Assessment

The dataset schema proposed to organize setplays into clusters is based on fuzzy clustering as stated in our previous work [22]. This section assesses if the schema presented meets the requirements to organize the full dataset of setplays into several groups. The main requirement is to split the dataset into clusters containing setplays that are semantically equivalents and can be used in the same game situation. In this work, we use a dataset containing ten times more instances than we did in our previous work. The contents of the dataset are also different. The setplays in this new dataset include all enhancements described in Sect. 3. We use, for the first time, the demonstration mode from Roboviz to integrate it to SPlanner. This assessment applies to our complete proposal for generating an organized dataset to support the learning of setplays from demonstration (see Fig. 19).

In Sect. 6.1, we describe our experimental setup. Section 6.2 describes Fuzzy Silhouette (FS) and the methods we have used to calculate FS as our main metric to assess the eligibility of our cluster organization.

6.1 Experimental setup

One of the goals when using FCM is to define the best number of clusters to adequately represent the system of interest. A widely known and simple scheme for defining

Fig. 20 Schematic representation of the dataset schema. The first level contains the features that identify the setplays. The second level contains the features that describe the steps of each setplay

the number of clusters consists of executing the fuzzy clustering algorithm several times for different numbers of clusters and then selecting the particular number of clusters that provide the best result according to a specific criterion [2, 12].

Many Cluster Validation Indexs (CVIs) were proposed and analyzed in recent works [8, 9]. These CVIs are used to assess the quality of data organization into clusters. CVIs are popular measures to assess the number of clusters used in a particular data organization. We use FS to verify if our proposed data schema can provide a good organization for setplays and represent the concept of semantic equivalence in an adequate number of clusters. FS nonmonotonic bias, good scalability to large datasets, and low computation costs are the main reasons for our choice [5].

For the assessment, we used the FCM algorithm to split a sample dataset into clusters. The dataset was generated using Roboviz demonstration mode and SPlanner. The dataset was composed of 181 setplays. To provide diversity, the setplays were created in six different play-modes: play on, goal kick, kick in, kick-off, free kick and corner kick. For all play-modes, we generated both offensive and defensive setplays. At least four different setplays were created for each play-mode. Hence, the dataset is composed both of simple and complex setplays.

We adapted the standard FCM implementation [3] to use the norms defined in our previous work [22]. The algorithm was also developed to split the clustering process into two stages as described. At the first stage, the feature *stepsList* of the setplays objects is ignored. When the centroids of each cluster are defined, the *abortCond* feature is assigned the

binary tree from the *abortCond* property of the instance with a higher membership value in the fuzzy partition matrix.

We populated our dataset with setplays created from real game situations extracted from Roboviz. We thus considered $C^{(1)} = \sqrt{N}/2, \ldots, 2 \times \sqrt{N}$; where N is the number of instances in the dataset; to run the FCM and to find different organizations for the dataset. FS were used to assess each setup with different values of c. For each value of c, we ran 10 instances of the FCM algorithm initializing with a random prototype of clustering. The higher value of FS among the 10 instances was considered the representative value for that c particular instance.

After running the experiment for the first stage we found the optimal value $C^{(1)^*}$ for $C^{(1)}$. We used a cluster instance for $C^{(1)} = C^{(1)^*}$ to start the second stage.

To define the membership of dataset instances for each of the $C^{(1)^*}$ clusters, consider $\Delta\mu_i = \left(\max \mu_{i,j}\right) - \gamma$; $j = 1, \ldots, N; i = 1, \ldots, C^{(1)^*}$; where $\gamma \in [0, 1]$ is a constant used to define how flexible the membership condition was. Greater values of γ tend to give more flexibility to the membership condition, i.e., instance of setplays with membership degrees far from the best-valued instance were also considered as a member of that particular cluster. When γ tends to 1, all the instances of setplays in the dataset were considered members of the *ith* cluster. When γ tended to 0 only the best-valued instance was considered a member of the *ith* cluster. The clustering setup degenerated to a set of singletons, i.e., groups with only one instance.

We defined that the *jth* instance was a member of the *ith* cluster if $u_{i,j} > \Delta\mu_i$. In this work, we used $\gamma = 0.5$ for all the experiments.

6.2 Fuzzy Silhouette (FS)

Consider the fuzzy partition matrix $P = \left[\mu_{i,j}\right]_{CxN}$; where C is the number of clusters used in the FCM algorithm, N is the number of objects to be organized into clusters, $\mu_{i,j}$ is the membership degree of object j to cluster $i, i = 1, \ldots, C$, $j = 1, \ldots, N$. The FS is defined by [5]:

$$FS = \frac{\sum_{j=1}^{N} \left(\mu_{p,j} - \mu_{q,j}\right)^{\alpha} s_j}{\sum_{j=1}^{N} \left(\mu_{p,j} - \mu_{q,j}\right)^{\alpha}}, \tag{1}$$

where $\mu_{p,j}$ and $\mu_{q,j}$ are the first and second largest elements of the *jth* column of the fuzzy partition matrix, respectively, and α is a user-defined weighting coefficient. s_j is the silhouette of object j defined as follows:

$$s_j = \frac{b_{p,j} - a_{p,j}}{\max\left(a_{p,j}, b_{p,j}\right)}, \tag{2}$$

where $a_{p,j}$ is the average distance of object j to all other objects *belonging* to cluster p. The distance is calculated using the norms defined in our previous work [22]. *Belonging* to cluster p means that the membership of the *jth* object to the *pth* fuzzy cluster, $\mu_{p,j}$, is higher than the membership of this object to any other fuzzy cluster, i.e., $\mu_{p,j} > \mu_{q,j} \forall q \in \{1, \ldots, c\}, q \neq p$. Let $d_{q,j}$ be the average distance object j to all objects *belonging* to another cluster q, $q \neq p$. $b_{p,j}$ is the minimum $d_{q,j}$ computed over $q = 1, \ldots, c$, $q \neq p$. Exponent α is an optional user-defined parameter ($\alpha = 1$, by default). When α approaches zero, the *FS* measure defined in (1) approaches the *Crisp* Silhouette (CS) measure which serves as a basis to define FS [5]. CS is the *crisp* counterpart of FS and is used for *crisp* clustering algorithms applied to hard datasets whereby no overlap between clusters is present. Conversely, increasing α moves FS away from CS by diminishing the relative importance of data objects in overlapping areas. Accordingly, increasing α tends to stress the effect of revealing smaller regions with higher data densities (sub-clusters), if they exist. Such an effect can be particularly useful, for example, when dealing with data sets contaminated by noise. Bearing this in mind, exponent α can be seen as an additional tool for exploratory data analysis, as is the case with fuzzifier exponent (m) of the FCM algorithm. From another perspective, the FS with exponent α can be seen as a family of parameterized CVIs, rather than a single measure with a coefficient that must be adjusted to a specific problem in hand. We here used ($m = 2, \alpha = 2$), since it proved to be the parameters which better explore the overlapping feature between clusters in our dataset [22].

FS is a maximization CVI. Hence, the higher the value of FS for a particular value of c, the better this specific clustering is. The goal when using FS to evaluate the clusters setup generated by FCM is to find the value of c that maximizes FS.

6.3 Results

Our first experiment consists in running the FCM algorithm to organize the first level of our dataset. The goal of this experiment is to define the appropriate number of clusters (C) we should use for this dataset in its first level. Figure 21 shows the results.

The best value for FS was found when $C^{(1)} = 27$. We can observe that after $C^{(1)} = 27$ the value of FS tends do decrease. We can choose $C^{(1)^*} = 27$ as the number of clusters we use to organize our dataset in the first level. We can also check the number of instances per cluster when we consider $C^{(1)} = 27$. For all clusters $c^{(1)} = 1, \ldots, C^{(1)}$, the number of setplays on each cluster is shown in Fig. 22.

We can see the most of the clusters contains between 4 and 6 setplays. The largest group is the cluster $c^{(1)} = 11$ with 10 instances. There are only 5 singletons (cluster with a single instance).

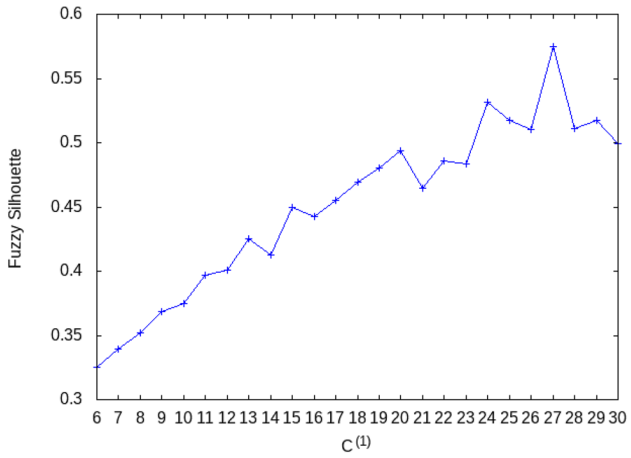


Fig. 21 Results of Fuzzy Silhouette for level 1 of the dataset. $C^{(1)}$ varies from 6 to 30

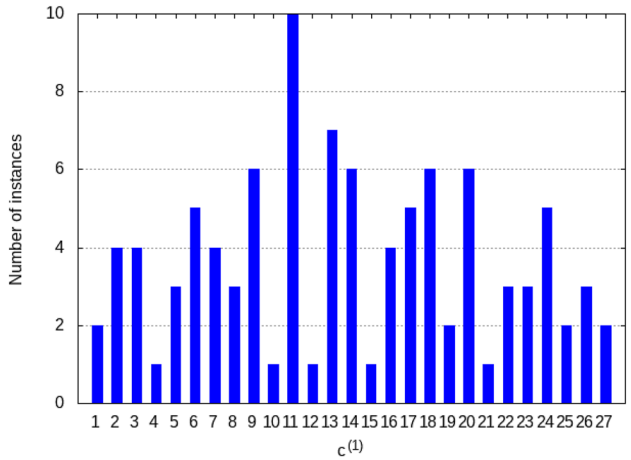


Fig. 22 Number of instance on each cluster after level 1 organization considering $C^{(1)^*} = 27$

This result is appropriate because the existing selection method in the FSF uses a CBR strategy. This solution has proved to generate good results for small groups of setplays.

In the second experiment we run the FCM algorithm over each of the 27 clusters found in level 1 organization. We consider that singletons are not eligible for a subdivision in level 2. The cluster where $2 \times \sqrt{(N_{c^{(1)}})} \leq \sqrt{(N_{c^{(1)}})}/2$ can not also be subdivided in level 2.

In Fig. 23, we see the results of FS for the subdivision of eligible clusters from level 1. In all cases, the best values of FS emerge when $C^{(2)} = 2$. Only in two cases, the value of FS for $C^{(2)} = 3$ is similar to the value obtained for $C^{(2)} = 2$. These results show that most of the eligible clusters from level 1 can be successfully split into two sub-clusters in level 2.

6.4 Discussion

We demonstrated that it is possible to split a large dataset into clusters containing a small number of setplays. For those groups with not so few setplays, we can run the second level of clustering to get sub-groups of instances. The main reason why we need to work with small groups is that the simulation is a soft real-time application. Robots are required to take new decisions on each 20 ms. If we use a large dataset with no clustering, the selection method, using CBR, would spend too much computational time and make agents lose the simulation cycle.

Our results demonstrate that it is feasible to find a fuzzy clustering organization that reduces the number of instances on each cluster. This way, we can consider using larger datasets to feed the reinforcement learning engine described in our future work. Our results confirm that we can use the new features added to RoboViz and SPlanner (see Sects. 3, 4) to generate the instances for the dataset. The new data could be organized and provide an eligible dataset for the last stage of the LfD approach we use in this project.

As far as we know, there is no previous work that presented a strategy to generate a large dataset from soccer domain experts and organize it so that a robotic soccer team can use it in real-time to learn coordinated strategies adapting these plans to the skills of each robot. Hence, we can consider our results as a novel contribution o state the art in this field.

7 Conclusion and future work

This paper presented the complete process of generating a dataset for learning setplays from demonstration in a SSIM3D team. The process uses some tools such as RoboViz, SPlanner, and FSF. The final step is an organizer which uses the FCM algorithm in two levels to organize the dataset.

Some enhancements were introduced to SPlanner, adapting it to the SSIM3D teams' needs to attend to the requirements of this process. The proposed improvements in SPlanner allow setplay's designers to create more sophisticated plays using passes. The *passToBestPlayer*, for instance, enables the creation of multi-flow setplays, which delegate to the team's behavior implementation to decide which teammate is the best player to receive the pass. Using a multi-flow setplay, the designer can foresee all possible flows of the play, considering each player selected to act as a pass receiver by the team's algorithm. The updated *intercept* action allows sending the target region coordinates to the intercept command, making it easier for the teams' developers to implement this action.

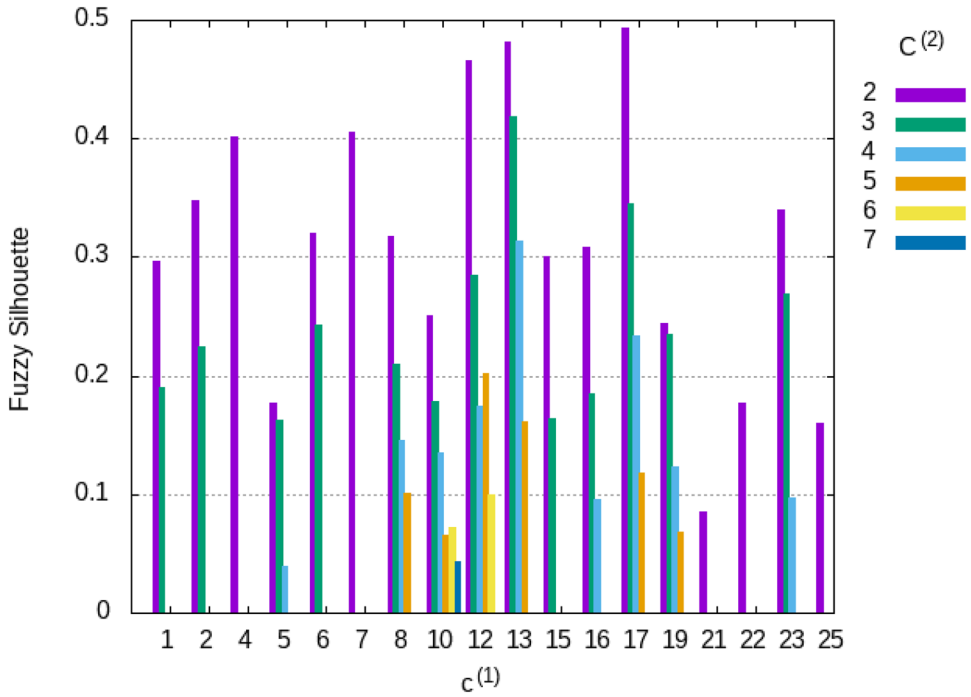


Fig. 23 Fuzzy Silhouette for level 2 sub-clusters for each cluster $c^{(1)}$

When performing the intercept behavior, the player who receives the pass can start moving to the target region even before his teammate can complete the pass. Another essential behavior proposed here was the *offensive marker* behavior. After all, in humanoid robots soccer, the kick is not an instantaneous action because robots need to move lots of joints to perform a complete movement. In this sense, the teams must consider the time to prepare and execute a pass before an opponent can intercept it. The new *offensive marker* can block the opponent player in advance and stop its movement towards the teammate who is performing a pass. This behavior allowed the use of the strategic plan to enable the execution of more passes. All these scenarios were validated using BahiaRT as a case study.

To complete the set of strategic plans for a soccer team, offensive and defensive setplays were also included. These options are now enabled in SPlanner through the new proposed behaviors: *defensive marker* and *become owner*. The *defensive marker* allows the designer to define actions for the team's players to mark the opponents blocking their dribbling and passing actions and preventing them from receiving passes. The *become owner* is used when one of the team's players is closer enough to the opponent with ball possession to regain it.

Another contribution of this work is the design of more realistic setplays using SPlanner. SPlanner is now more suitable for use in humanoid soccer games, easing the use of this toolkit for the SSIM3D competition and other humanoid robotic soccer leagues. From a practical viewpoint, it is essential to turn these tools closer to experts' strategies, considering the demonstrations made both by humans and robotic soccer experts.

Some changes in RoboViz were also necessary. A new demonstration mode was introduced, enabling users to capture a specific game situation to be used by SPlanner to start a recent demonstration. This integration is essential to turn the complete toolkit usable by domain experts.

The results presented here make the SPlanner a more suitable tool for the design of more realistic moves closer to human soccer. The tool produced in this work provides a familiar interface for soccer specialists. By including both soccer specialists and robotic soccer specialists among the advisors, the possibility of generalizing specialized knowledge through the LfD engine is expanded. The main idea is to use the modified SSIM3D viewer (RoboViz) and the new version of SPlanner to allow lots of soccer domain experts to populate a dataset with demonstrations of setplays. This dataset could be used in situations they have seen in previous games' logs. This paper's contributions turn the soccer strategies closer to real soccer situations and allow a more significant number of experts to contribute to our LfD project.

The last tool presented in this work is the dataset organizer. It uses a 2-level clustering engine to split the data set into groups of setplays. The membership is defined in a fuzzy fashion. This way, we can see some overlap between clusters. This situation reflects the data's nature that composes the setplays, once it is not easy to classify the setplays using crisp criteria.

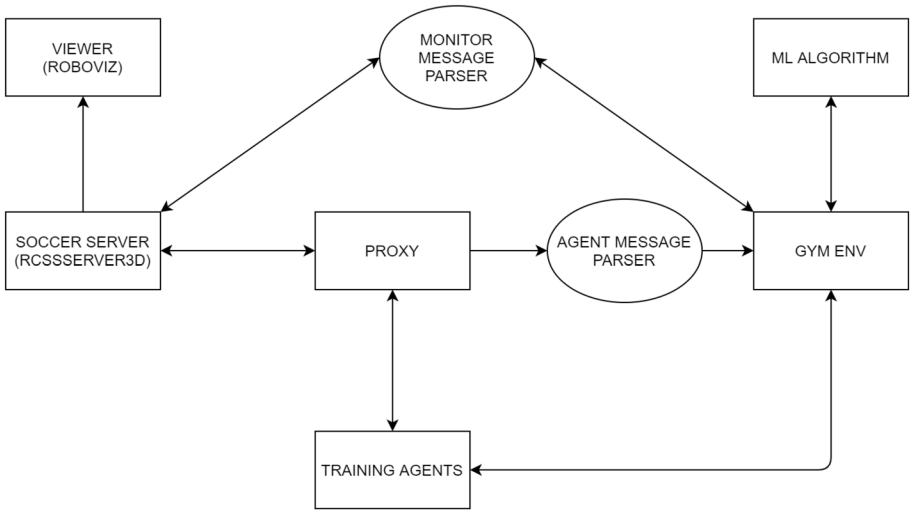


Fig. 24 Schematic representation of Open AI Gym environment for RoboCup Soccer Server 3D Simulator

Future work includes populating the dataset, gathering demonstrations from many soccer experts, and test our LfD engine to prove the efficacy of this approach to create high-level strategies for MAS. Finally, we intend to run a new set of tests of setplays with further improvements in the BahiaRT.

As an undergoing initiative, we are working in a single environment integrating OpenAI GYM [4], RoboCup Soccer Server3D Simulator, and RoboViz (Fig.24). A proxy intermediates communications between the Soccer Server, OpenAI Gym, and Training Agents modules to avoid modifying the agent's effectors. OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. The latest developments by other 3D soccer simulation teams have used the OpenAI Gym. It enables us to create a wide custom environment shared with the soccer 3D community in the future. The Gym Env module will be the bridge between Machine Learning Algorithms and Training Agents modules, allowing the Training Agents module to receive input from the OpenAI GYM to optimize the team's tactics player's behaviors. The Agent Message Parser is responsible for collecting and parse all the simulator's perceptions to agents' sensors and feed the Gym Env World Model. This World Model contains all information collected from the simulator. This information is the basis of the learning model. The Monitor Agent Parser collects noiseless information from the simulator. The Soccer Server sends the same information it sends to a viewer (e.g., RoboViz) to the Monitor Agent Parser. The Monitor Agent Parser also can send some commands as a Trainer, so the training episodes situation can be set up and repeated as many times as the learning strategy defines.

The MAS developers can use this training environment to allow the MAS to learn a policy to select an appropriate cluster of setplays to be used in a given game situation. Both cluster selection and the choice of setplay within the selected cluster can be fast to enable agents to take decisions limited to the simulation cycle of 20 ms.

Other future works include assessing this proposal in other domains than robotic soccer. We plan to adapt SPlanner and the other tools to a domain of oil platforms inspection with a MAS composed by Unmanned Aerial Vehicles (UAVs). The main idea is to demonstrate that our proposal can be applied to any MAS where the domain expert's intuitive knowledge must be extracted and transferred to the agents.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Almeida F, Abreu PH, Lau N, Reis LP (2013) An automatic approach to extract goal plans from soccer simulated

matches. Soft Comput 17(5):835–848. https://doi.org/10.1007/s00500-012-0952-z
2. Babuška R (2012) Fuzzy modeling for control, vol 12. Springer, Berlin
3. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers, Norwell
4. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI gym. CoRR arXiv:abs/1606.01540
5. Campello R, Hruschka E (2006) A fuzzy extension of the silhouette width criterion for cluster analysis. Fuzzy Sets Syst 157(21):2858–2875. https://doi.org/10.1016/j.fss.2006.07.006
6. Cravo J, Almeida F, Abreu PH, Reis LP, Lau N, Mota L (2014) Strategy planner: graphical definition of soccer set-plays. Data Knowl Eng 94:110–131. https://doi.org/10.1016/j.datak.2014.10.001
7. D'Ambrosio DB, Stanley KO (2013) Scalable multiagent learning through indirect encoding of policy geometry. Evolut Intell 6(1):1–26. https://doi.org/10.1007/s12065-012-0086-3
8. Eustáquio F, Camargo H, Rezende S, Nogueira T (2018) On fuzzy cluster validity indexes for high dimensional feature space. In: Kacprzyk J, Szmidt E, Zadrożny S, Atanassov KT, Krawczak M (eds) Advances in fuzzy logic and technology 2017, advances in intelligent systems and computing. Springer, Berlin, pp 12–23
9. Eustáquio F, Nogueira T (2018) On monotonic tendency of some fuzzy cluster validity indices for high-dimensional data. In: 2018 7th Brazilian conference on intelligent systems (BRACIS), pp 558–563. https://doi.org/10.1109/BRACIS.2018.00102
10. Fabro JA, Reis LP, Lau N (2014) Using reinforcement learning techniques to select the best action in setplays with multiple possibilities in Robocup soccer simulation teams. In: 2014 joint conference on robotics: SBR-LARS robotics symposium and robocontrol, pp 85–90. IEEE, Sao Carlos, Sao Paulo, Brazil. https://doi.org/10.1109/SBR.LARS.Robocontrol.2014.47. http://ieeexplore.ieee.org/document/7024261/
11. Freelan D, Wicke D, Sullivan K, Luke S (2014) Towards rapid multi-robot learning from demonstration at the RoboCup competition. In: RoboCup 2014: robot World Cup XVIII. Lecture notes in computer science, pp 369–382. Springer, Cham. https://doi.org/10.1007/978-3-319-18615-3_30
12. Höppner F, Klawonn F, Kruse R, Runkler T (1999) Fuzzy cluster analysis: methods for classification, data analysis and image recognition. Wiley, New York
13. Liemhetcharat S, Veloso M (2017) Allocating training instances to learning agents for team formation. Auton Agents Multi-Agent Syst 31(4):905–940. https://doi.org/10.1007/s10458-016-9355-3
14. Micalizio R, Torta G (2016) Explaining interdependent action delays in multiagent plans execution. Auton Agents Multi-Agent Syst 30(4):601–639. https://doi.org/10.1007/s10458-015-9298-0
15. Mota L, Lau N, Reis LP (2010) Co-ordination in RoboCup's 2D simulation league: setplays as flexible, multi-robot plans. In: 2010 IEEE conference on robotics, automation and mechatronics, pp 362–367. https://doi.org/10.1109/RAMECH.2010.5513166
16. Mota L, Reis LP, Lau N (2011) Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics 21(2):434–444. https://doi.org/10.1016/j.mechatronics.2010.05.005
17. Panella A, Gmytrasiewicz P (2017) Interactive POMDPs with finite-state models of other agents. Auton Agents Multi-Agent Syst 31(4):861–904. https://doi.org/10.1007/s10458-016-9359-z
18. Ramos CEdR (2017) Planejador Multiagentes para Criaçã de Jogadas Ensaiadas em um Time de Futebol de Robôs Simulados. Bachelor Thesis, Universidade do Estado da Bahia (UNEB), Salvador, Bahia, Brazil (2017). Published: Bachelor Thesis, Universidade do Estado da Bahia (UNEB), supervised by Marco A. C. Simões
19. Shi H, Lin Z, Hwang K, Yang S, Chen J (2018) An adaptive strategy selection method with reinforcement learning for robotic soccer games. IEEE Access 6:8376–8386. https://doi.org/10.1109/ACCESS.2018.2808266
20. Simoes MAC, Nobre J, Sousa G, Souza C, Silva RM, Campos J, Souza JR, Nogueira T (2020) Strategy planner: enhancements to support better defense and pass strategies within an LfD approach. In: 2020 IEEE international conference on autonomous robot systems and competitions (ICARSC), pp 46–52. IEEE, Ponta Delgada, Portugal. https://doi.org/10.1109/ICARSC49921.2020.9096188. https://ieeexplore.ieee.org/document/9096188/
21. Simões MAC, Nogueira T (2018) Towards setplays learning in a multiagent robotic soccer Team. In: 2018 Latin American robotic symposium, 2018 Brazilian symposium on robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), pp 277–282 (2018). https://doi.org/10.1109/LARS/SBR/WRE.2018.00058
22. Simões MAC, da Silva RM, Nogueira T (2020) A dataset schema for cooperative learning from demonstration in multi-robot systems. J Intell Robot Syst 99(3–4):589–608. https://doi.org/10.1007/s10846-019-01123-w
23. Wooldridge M (2009) An introduction to multiagent systems, 2nd edn. Wiley, Chichester
24. Yu C, Zhang M, Ren F, Tan G (2015) Multiagent learning of coordination in loosely coupled multiagent systems. IEEE Trans Cybern 45(12):2853–2867. https://doi.org/10.1109/TCYB.2014.2387277
25. Zhang C, Sinha A, Tambe M (2015) Keeping pace with criminals: designing patrol allocation against adaptive opportunistic criminals. In: Proceedings of the 2015 international conference on autonomous agents and multiagent systems. Istanbul, Turkey, pp 1351–1359
26. Zhang C, Tambe M (2015) Modeling, learning and defending against opportunistic criminals in urban areas (Doctoral Consortium). In: Proceedings of the 2015 international conference on autonomous agents and multiagent systems, pp 1971–1972. Istambul, Turkey
27. Zhang S, Jiang Y, Sharon G, Stone P (2017) Multirobot Symbolic Planning under Temporal Uncertainty. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp 501–510. São Paulo, Brazil
28. Zhou J, Purvis M, Muhammad Y (2015) A combined modelling approach for multi-agent collaborative planning in global supply chains. In: 2015 8th international symposium on computational intelligence and design (ISCID), vol 1, pp 592–597. https://doi.org/10.1109/ISCID.2015.13