robotics

MDPI

Article

Probability-Based Strategy for a Football Multi-Agent Autonomous Robot System

António Fernando Alcântara Ribeiro [1], Ana Carolina Coelho Lopes [1], Tiago Alcântara Ribeiro [2], Nino Sancho Sampaio Martins Pereira [3], Gil Teixeira Lopes [4] and António Fernando Macedo Ribeiro [2,*]

[1] Industrial Electronics Department, University of Minho, 4800-058 Guimarães, Portugal; pg47028@alunos.uminho.pt (A.F.A.R.); pg50176@alunos.uminho.pt (A.C.C.L.)
[2] Industrial Electronics Department, ALGORITMI Centre, 4800-058 Guimarães, Portugal; id9402@alunos.uminho.pt
[3] Dyson Ltd., 86 Hullavington Airfield, Hullavington, Chippenham SN14 6GU, UK; nino.pereira@dyson.com
[4] INESC TEC, Business Science Department, University of Maia, 4475-690 Maia, Portugal; alopes@umaia.pt
* Correspondence: fernando@dei.uminho.pt

Abstract: The strategies of multi-autonomous cooperative robots in a football game can be solved in multiple ways. Still, the most common is the "Skills, Tactics and Plays (STP)" architecture, developed so that robots could easily cooperate based on a group of predefined plays, called the playbook. The development of the new strategy algorithm presented in this paper, used by the RoboCup Middle Size League LAR@MSL team, had a completely different approach from most other teams for multiple reasons. Contrary to the typical STP architecture, this strategy, called the Probability-Based Strategy (PBS), uses only skills and decides the outcome of the tactics and plays in real-time based on the probability of arbitrary values given to the possible actions in each situation. The action probability values also affect the robot's positioning in a way that optimizes the overall probability of scoring a goal. It uses a centralized decision-making strategy rather than the robot's self-control. The robot is still fully autonomous in the skills assigned to it and uses a communication system with the main computer to synchronize all robots. Also, calibration or any strategy improvements are independent of the robots themselves. The robots' performance affects the results but does not interfere with the strategy outcome. Moreover, the strategy outcome depends primarily on the opponent team and the probability calibration for each action. The strategy presented has been fully implemented on the team and tested in multiple scenarios, such as simulators, a controlled environment, against humans in a simulator, and in the RoboCup competition.

Keywords: football strategy; multi-agent; autonomous robot; RoboCup; probability-based; play generator; decision trees; heat maps; simulation

Citation: Ribeiro, A.F.A.; Lopes, A.C.C.; Ribeiro, T.A.; Pereira, N.S.S.M.; Lopes, G.T.; Ribeiro, A.F.M. Probability-Based Strategy for a Football Multi-Agent Autonomous Robot System. *Robotics* **2024**, *13*, 5. https://doi.org/10.3390/robotics13010005

Academic Editors: Charalampos P. Bechlioulis and Kagan Eugene

Received: 8 November 2023
Revised: 18 December 2023
Accepted: 20 December 2023
Published: 23 December 2023

1. Introduction

RoboCup started in 1997 [1] and leagues have become more competitive over the years based on the motivation of all teams to research and develop. The "Middle Size League" (MSL) is no exception. It represents one of the primary football leagues and encompasses numerous research areas. Artificial intelligence, multi-agent systems, computer vision, communication systems, strategy, decision-making, control systems, power electronics, mechanical engineering, and many more are the foundations of MSL. Having robots capable of playing a full game is a requirement. The University of Minho first participated in MSL in 1999 with the MinhoTeam, now known as LAR@MSL [2].

One of the RoboCup goals is to defeat the world champion human team in soccer with the best team by 2050 [1,3]. Teams with five fully autonomous robots play against each other and are free to design their hardware and software while complying with the league rules (Figure 1). These rules are an extension of FIFA's main soccer rules adapted to the needs of this robotic competition. Most MSL teams use a similar strategy architecture

based on how humans approach the problem, which is called "Skills, Tactics and Plays" (STP). This paper presents a completely new strategy approach designed to generate and decide the game outcome and players' intentions in real-time. It is highly dynamic and flexible to different game situations since it bases all the calculations on live real-time data acquired from the robots. This new solution allows playing and maintaining synchronism between all the robots without any previous game experience, neither predefined plays nor data from previous games. This solution can even be applied in simulation games because the outcome of the strategy is a series of simple instructions and those can be sent to a real-world game or to a simulation game. The necessity for a more calculated solution, developed without reliance on pre-existing data, was significant. Additionally, time constraints were a critical factor, not only in development but in between and during games. Time is very limited between games, and the parametrization of the strategy to a different opponent is crucial. Using a playbook requires time to plan and compile, also meaning that the larger the playbook, the more complex it gets, and more time is needed to edit/change/parameterize, making it extremely difficult between and during games.

Figure 1. Game situation in RoboCup 2023.

2. Related Work

The predominant approach to addressing strategic challenges in robot football typically involves a playbook-based strategy. This implies that most plays must be meticulously pre-conceived and developed beforehand to apply them to various game situations subsequently.

2.1. Most Common Used Architecture

The "Skills, Tactics, and Plays (STP)" architecture was initially developed by the team CMDragons [4–6], a "Small Size League" team, with the primary goal of coordinating a full team of robots with the main focus of scoring goals. It was later adapted and used by the MSL team, Tech United Eindhoven [7] to overcome their main issues with the algorithms used at the time. As the name suggests, the STP is divided into three parts: Skills, Tactics, and Plays. This section explains how each component works and how they generate the desired output when working together.

2.1.1. Skills

The most basic actions that a robot can take are designated as Skills, and even though they vary from team to team, the amount and complexity of these skills is low. Skills are not oriented to the main goal of scoring; nonetheless, they are essential actions that robots perform to evolve and contribute to the game. Some available skills include moving

around the field at a certain speed, avoiding obstacles, controlling the ball with the dribbles, rotating with the ball, and shooting the ball [7].

2.1.2. Tactics

Tactics are a group of skills in a certain order with an individual robot objective. Tactics are given to every robot, regardless of whether they are part of the main team plan or play. Each robot has a different set of tactics based on the game state. These are usually implemented as finite-state machines that provide a sequence of actions and skills to use within each tactic. Tactics within this framework are also parameterized. Examples include defending the goal, executing a pass, intercepting the opponent's pass, and assisting in an attack [4–6].

2.1.3. Plays

Building upon the tactics, the plays delineate the specific roles and sequential actions of each robot. A play is a team plan that defines a group of tactics for each robot on the team, and the group of tactics is given in a specific sequence. Also, part of the play is defined by the conditions under which that play must be used, as well as the exit conditions specifying when to transition out of the play. All the plays are compiled into a playbook that can be modified to change the robot's behavior in different game situations. A play is activated as soon as the exit conditions of the preceding play are met, and it continues to be in effect until its own exit conditions are likewise fulfilled.

2.1.4. Playbook

As a rule of thumb, teams have multiple playbooks so that they can have different strategies against different opponents. During a game, no human is allowed to touch the team's main computer or base station, and so, during the game, only one playbook is used. Playbooks can vary in specificity, from very detailed to broadly generalized, depending on a team's needs and preferences. If one assumes that a single play covers all possible game states, then the resulting playbook would consist of just that one play. Realistically, this is not the case. When a play has multiple outcomes and needs different plans based on a specific game variable, these are divided into two separate plays that are chosen based on that game variable. The choice of play can also be influenced by its weight associated with a variable and not just game state variables. This happens because, in the same game situation, there are multiple solutions, and one can choose the play based on a predetermined weight [7].

2.2. Machine Learning Approaches

Diverse approaches were employed in developing the skills by other teams, such as utilizing reinforcement learning [8,9]. However, the decision-making aspect continued to rely on traditional methods within the STP architecture [10]. Because of the STP dependence, there must still be a previously created playbook that addresses various potential game situations.

Some machine learning algorithms were also used to transition between different plays or strategies. For example, a Graph Evolution, developed in 2014 [11], implements a new management method, which is the current play being used and how to transition into other plays on a probability basis, but the dependency of a playbook persists.

2.3. Different Play Transition Methods

The choice of which play to choose given a particular game state was also solved with Bayesian classifiers in 2018 [12] by a Brazilian small-size league team. However, the problem of being required to think ahead of all the different plays possible and creating a playbook is still in place.

The use of probabilities for plays has been used by the team RoboJackets [13], but its implementation was based on parameterizing and deciding the desired play and not generating them in real-time.

2.4. Alternatives to STP

The MuJoCo team implemented an approach that diverges from the STP architecture, opting instead for a solution entirely based on reinforcement learning [14], applied within a simulation environment. The downside of having a full reinforcement learning implementation [15] is the lack of modularity and parameterization, which are important when playing against different teams in a short time span. When a new opposing team employs different behaviors, reinforcement learning algorithms can lack flexibility and speed when adopted. Implementing changes often requires more time and computational resources compared with hand-coded strategies. Classic strategies have the advantage of allowing behavior changes on the fly during a match (during the break) or in between matches. Some simulation league teams use genetic algorithms [16], which do not rely on the STP architecture. However, these have other limitations, such as modularity constraints and training requirements.

3. Robots Hardware

To help understand the development environment for the strategy, the robot hardware and mechanical description are presented in Figure 2.

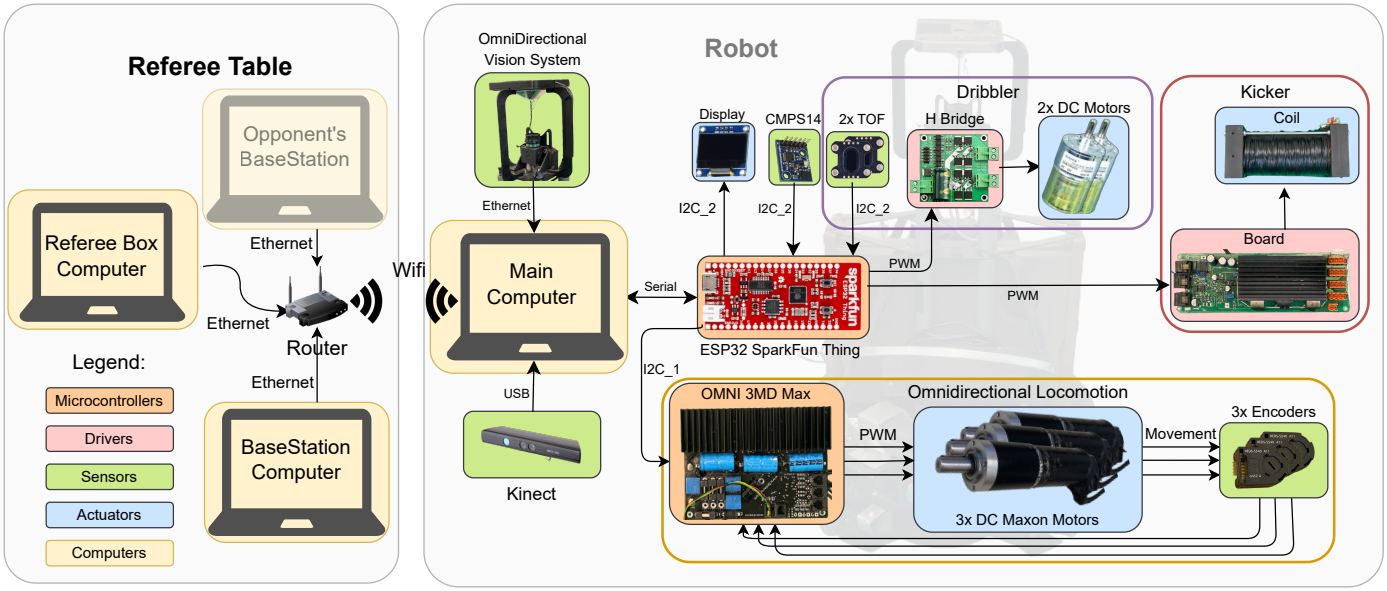


Figure 2. Robot hardware diagram.

The LAR@MSL team has five robots: four strikers and a goalkeeper. All the strikers have the same hardware and software, but the solution for the goalkeeper to optimize its objectives is slightly different. The differences between the strikers and the goalkeeper are the position of the Kinect Depth Camera [17], the upwards-extending arm, and the addition of a LiDAR sensor at the back of the robot. Figure 3 depicts a striker robot and its components.

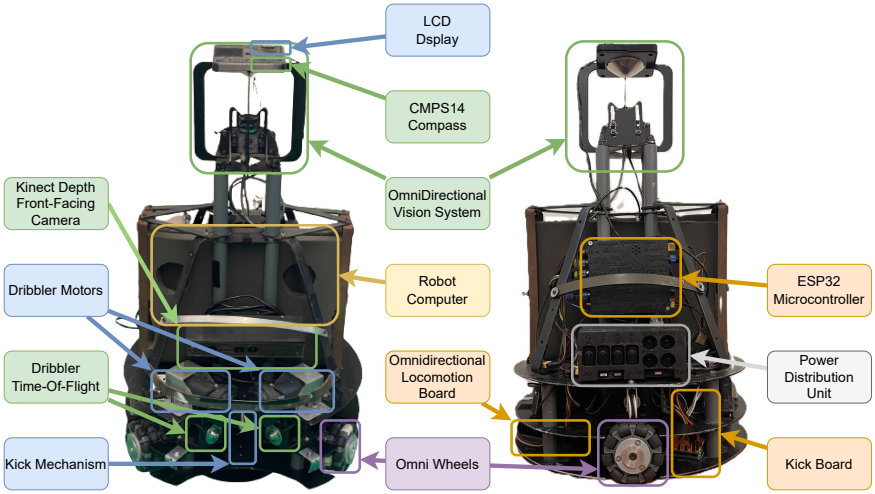
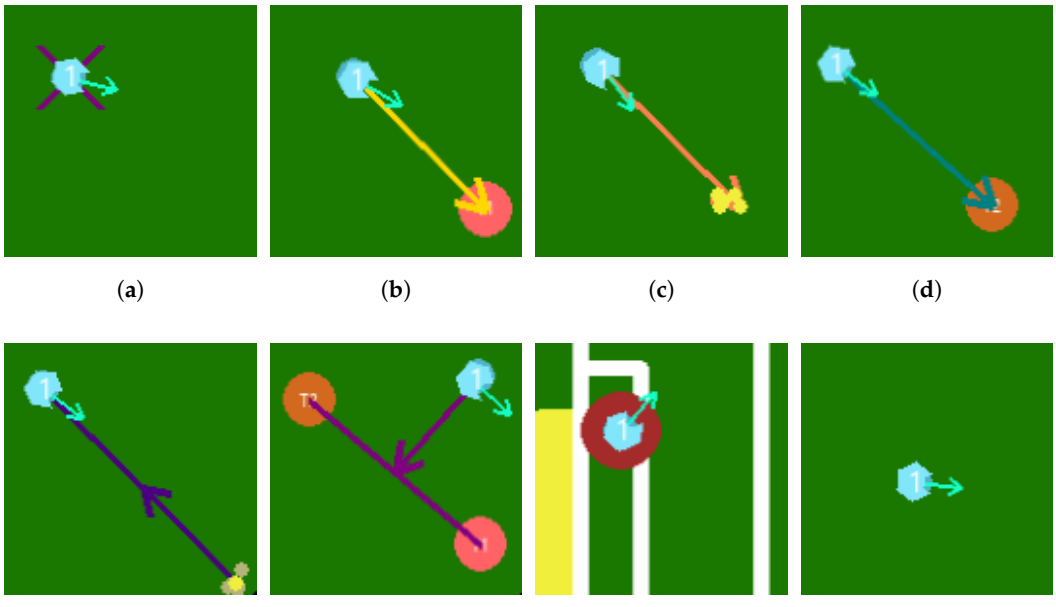


Figure 3. Robot photo with the main elements highlighted (front and back views).

4. LAR@MSL Skills

Following the same logic as the STP, the robots can perform certain skills, and they are fully autonomous in that regard. For example, the robots can be instructed to follow the ball, and that is enough for the robots to detect the ball, obtain its precise location, calculate its trajectory, and move to grab it. Arguments are passed as a way to specify certain skill elements or to help the robot in a situation where it does not have the required information for the skill. Therefore, skills are the most fundamental actions that can be directed under this strategy. Specifically, in the LAR@MSL strategy [2], each robot can be assigned one of eight possible skills, each with up to seven arguments (Table 1). The number of arguments is higher than necessary at the moment, but it was planned to provide future expansion of the system. Figure 4 illustrates representations of all eight skills as visualized in the base station (main computer) interface. It provides helpful support to the plays section further on. Each robot is represented by a filled cyan hexagon containing the robot's number in the center, along with a cyan arrow indicating the robot's orientation. The ball is shown as a small yellow-filled circle.



(**a**) (**b**) (**c**) (**d**)

(**e**) (**f**) (**g**) (**h**)

Figure 4. Base station skill representation: (**a**) Stop. (**b**) Move. (**c**) Attack. (**d**) Kick. (**e**) Receive. (**f**) Cover. (**g**) Defend. (**h**) Control.

Table 1. Table of Skills and Arguments.

Skill	ID	A1	A2	A3	A4	A5	A6	A7
Stop	0	-	-	-	-	-	-	-
Move	1	X	Y	Ori	O_x	O_y	-	-
Attack	2	B_x	B_y	P	-	-	-	-
Kick	3	T_x	T_y	P/K	D_x	D_y	-	-
Receive	4	B_x	B_y	-	-	-	-	-
Cover	5	X_1	Y_1	X_2	Y_2	A	-	-
Defend	6	B_x	B_y	-	-	-	-	-
Control	7	D_x	D_y	Ori	P/K	-	-	-

4.1. Stop

The most basic skill has the purpose of stopping everything, hence the name **Stop**. It brakes the locomotion motors, stops the dribbling mechanism, and prevents the robot from kicking. It has no arguments, but it is used every time the game is stopped or in case of emergencies. Its representation is a purple cross below the robot.

4.2. Move

The **Move** skill, as the name suggests, is used to send a robot to a target location. It has five arguments, which are the location where the robot should go (x and y coordinates), a flag to define if the robot's orientation is free or locked, and the position the robot should orientate to (x and y). It is one of the most used skills and needs information from the localization system and obstacles in the path so that the robot can avoid them while performing the desired movement. The Move skill representation is a yellow arrow pointing to the location given by the arguments.

4.3. Attack

The **Attack** skill is responsible for trying to grab the ball as fast as possible. It receives the ball coordinates as a parameter but is only used by the robot if it does not see the ball by its own vision system. The parameter P is used for penalties, in which case the robot can move slower toward the ball so that it does not move out of place. An orange arrow towards the ball represents it.

4.4. Kick

When the robot intends to score a goal or pass the ball, the skill used is the **Kick**. The first two parameters indicate the position where the ball is supposed to be kicked. The third argument is a flag to indicate whether to kick or to pass, and the force applied is calculated accordingly. In a pass, it applies a specific force based on the distance to the teammate, but if the flag is to kick to the goal, the robot will decide the direction of the kick and use its maximum power. The kick is represented in Figure 4d by a blue arrow indicating the kick direction. The two final parameters are the required movement to achieve the best kicking position. Before the kick, the robot needs to rotate itself to the desired position. Taking advantage of its omnidirectional movement, the robot can move to a location to optimize and simplify the pass or kick to the goal. The base station calculates that position and takes into consideration the entire game and world states.

4.5. Receive

Paired with the kick skill, there is the **Receive** skill, which tells a robot that it is supposed to receive the ball coming towards it and should orientate itself and move accordingly. The only parameter required to receive the ball is the ball's position. If the ball is visible to the robot, it uses its own vision system data. Otherwise, it uses the parameter sent. It is represented by a dark blue arrow between the ball and the robot.

4.6. Cover

The **Cover** skill is mostly used in defensive situations. It has the purpose of positioning a robot between two specific coordinates that can be moving targets. For example, the robot can be instructed to cover point A from point B, and it calculates the desired coordinate based on those two arguments. As parameters, besides the two targets (X_1, Y_1) and (X_2, Y_2), the aggressivity of the player is sent, which consists of how much pressure a robot should put on one target or the other by a percentage. Translated to the real world, that percentage defines where on the line between the two targets the robot should position itself. In Figure 4f, the skill is represented with a purple line and a purple arrow. The line is drawn between the two targets, and the arrow points to the desired position calculated. In this example, the aggressiveness is 50%.

4.7. Defend

The **Defend** skill is used by the Goalkeeper, and the robot never leaves this skill. It is responsible for predicting the ball movement and arc trajectory, positioning it, and controlling the arms of the robot. It receives just two parameters consisting of the ball location. Just like in the attack and receive skills, it only uses them when the robot does not see the ball with its own vision system. It is represented with a brown circle.

4.8. Control

In demonstrations and test/debug situations (never during a game), this skill is used for the purpose of controlling the robot remotely by the user. It is controllable by a game controller connected to the base station computer or with a keyboard. This skill is also used in the simulation tool for controlling the opponents for testing purposes, and it has no representation since the robot is not autonomous in that situation. The parameters used are the desired movement in X and Y coordinates on the field, the orientation flag (whether it should orientate to the ball or not), and the pass or kick parameter for the robot to kick.

5. Strategy

RoboCup middle-size league teams have traditionally utilized strategies where robots make autonomous decisions [18]. This paper introduces a new strategy approach that relies on centralization, probability-based plays, and fast communication, redefining teamwork among robotic football players.

Unlike traditional methods, the LAR@MSL approach retains robot autonomy while taking advantage of a central computer, the base station, to provide skills, world information, and even ideal paths. This centralization enhances probability-based decision-making and synchronizes robot actions, creating an efficient and reliable strategy without a playbook. Besides the use of decision trees essentially for game and player state analysis, most of the decisions made in the strategy are probability-based, even the robot's positioning.

5.1. Architecture

Figure 5 represents the strategy architecture schematic used to implement the Play Generator, as well as the decision trees [19,20] and heat maps [21]. The most important modules are explained in the following sections. The schematic is also color-coded. The data fusion module, represented in dark blue, filters all the information gathered by the robots, creating a real-time world state map used throughout the strategy. The decision trees in red are responsible for part of the decision-making, both for the game and the player's decision trees. The Play Generator, shown in green, computes the optimal play for the current state of the game. Positioning and movement calculations fall within the purview of the heat maps and positioning modules, which are indicated in yellow. The two modules in purple, not discussed in this paper, are tasked with assembling the information, calculating the necessary arguments for packets, and transmitting them to the communication modules [22], thereby ensuring synchrony among the robots.

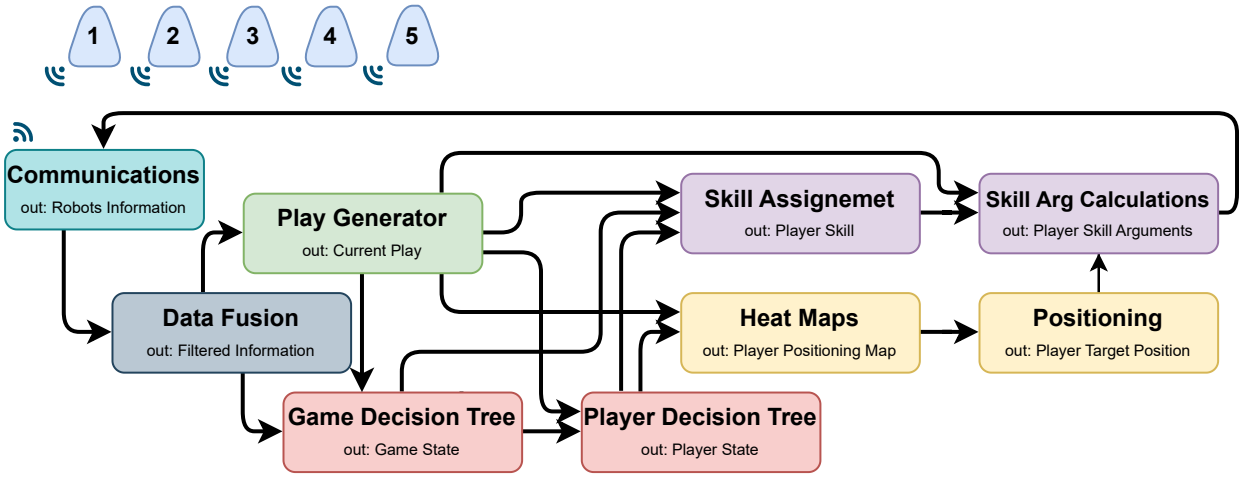


Figure 5. Strategy architecture schematic.

5.2. Play Generator

The Play Generator is a new method that generates the ideal desired play in real-time, without the need for predefined plays or a playbook. It uses the information gathered by the robots to calculate the ideal game situation outcome. It is modular and, therefore, it does not need to readjust for either the number of teammates or opponents. This method was designed for adaptable tuning to various opponent team styles, including speed-focused, precision-oriented, or hybrid approaches. The risk tolerance and safety measures [23] depend solely on how the probabilities assigned to potential plays shape overall tendency. An essential aspect of this system is that calibration and parameterization can be efficiently performed without the need for in-depth knowledge of football strategies or the formulation of potential plays. These processes are often time-intensive and reliant on specialized knowledge. Additionally, the algorithm leverages the homogeneity of the robots' capabilities; each robot is capable of performing any role. Predefined responsibilities do not dictate their actions. Still, they are determined by dynamic calculations of the most effective strategy to score a goal, contingent upon the current state of the game.

5.3. Probabilities

The probability of a pass or a goal is calculated based on the distance of the action and the opponent's influence. The action success probability is based on a custom function for testing purposes, which is modified later for a real-world transition. The distance between two robots, given their location, is first calculated as seen in Equation (1). Then, the probability of a successful pass without any opponent's influence is calculated in Equation (2). The meaning of the variables is presented in Table 2.

Table 2. Description of variables.

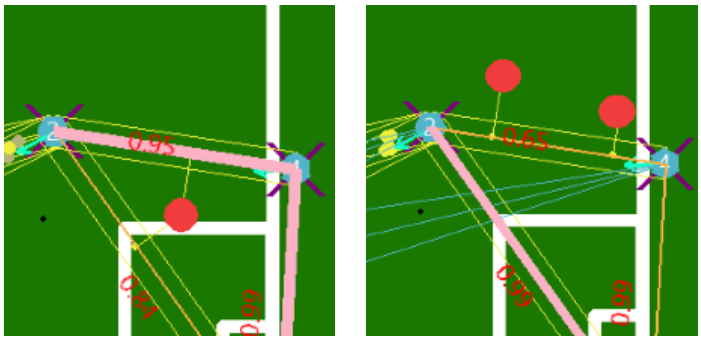
Variable Name	Meaning
$dist_{bR}$	Distance between robots
x_{ri}	Coordinate X from robot number i
y_{ri}	Coordinate Y from robot number i
p	Constant factor to normalize the Bayesian curve between [0, 1]
d	Bayesian curve deviation—margin used for the ideal action distance
$BDTA$	Best distance to action (pass or goal), ideal action distance between two robots, highest probability value
tp	totalProbability—action given probability
$prob$	totalProbability with the opponent's influence
$dist_{Opp\_to\_Line}$	Minimum distance from an opponent to the line of action (pass or goal)
$dist_{start\_infl}$	Maximum distance from where an opponent influences the action probability

$$dist_{bR} = \sqrt{(x_{r1} - x_{r2})^2 + (y_{r1} - y_{r2})^2} \quad (1)$$

$$tp = \frac{p}{d\sqrt{2\pi}} e^{\frac{-(dist_{bR} - BDTA)^2}{2d^2}} \quad (2)$$

The Equation (2) is based on a Bayesian curve [12,24]. The opponent's influence is applied by a multiplication factor based on the distance between the opponents and the line of pass being calculated, as shown in Equation (3). This influence is only calculated if the opponent robot is between the robots being analyzed and with a distance to the line of pass smaller than $dist_{start\_infl}$. Figure 6a shows an opponent affecting two lines, one of pass and one of goal, and Figure 6b shows an example of a line of pass being influenced by two opponents.

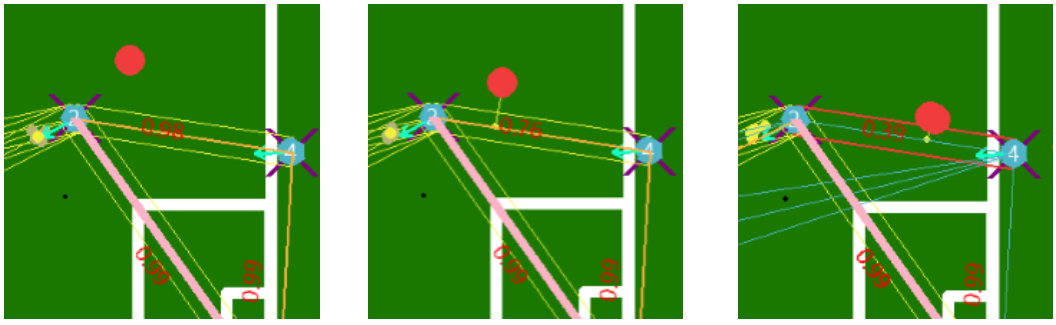
$$prob = tp * \frac{dist_{Opp\_to\_Line}}{S_{robot} * dist_{start\_infl}} \quad (3)$$



(**a**) (**b**)

Figure 6. Opponent's influence: (**a**) One opponent affecting two lines. (**b**) Two opponents affecting one line.

Figure 7a–c shows a possible pass with a high success probability and the influence of an opponent getting near the line of pass. In Figure 7c, the polygon surrounding the line of the pass is the limit of a possible pass. If the opponent intersects that boundary, the pass is not considered an available line.



(**a**) (**b**) (**c**)

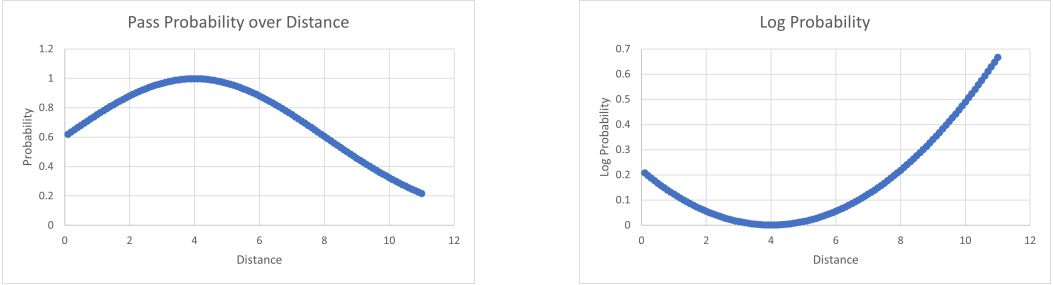
Figure 7. Pass probability with the opponent's influence based on its distance.

5.4. Log Probability

In order to create a map of probabilities between robots regarding possible actions, a transformation needs to occur into a workable value. After trying to find an algorithm able to determine the highest probability path, the best solution was to use the log probability [25] to translate probabilities into arbitrary values representative of map distances.

Log probability simply uses the logarithmic value of the probability in order to have an arbitrary value representative of its probability. The higher the probability, the smaller the distance. The formula to translate the probabilities is presented in Equation (4), with values constrained between 0 and 100. This upper limit serves to prevent computational errors and ensures the desired precision for operational efficacy. Figure 8 shows the Bayesian curve of an ideal pass probability over distance (a) and the respective log probability (b).

$$v = -log_{10}(prob) \tag{4}$$



(**a**) (**b**)

Figure 8. Log probability conversion: (**a**) Pass probability over distance. (**b**) Log probability of pass over distance.

The utilization of the log probability serves multiple purposes. From a computational standpoint, addition is significantly simpler and quicker than multiplication, an important consideration when evaluating different paths where all calculations are probability-based. For instance, the probability of events A and B occurring together would typically be calculated as A∗B. However, when applying logarithms, this multiplication is transformed into a sum, as outlined in [25]. The use of log probabilities is advantageous over floating points in terms of computer memory and accuracy, as the limits of the logarithm of a probability number between 0 and 1 vary between $]-\infty, 0]$ [25]. Therefore, the floating point resolution limitations do not exist.

5.5. Graph Generation and Path Finder

Regarding the graph that is generated based on the probabilities calculated, this is a complete graph [11,26], with n_{nodes} nodes, and the number of connections is given by Equation (5) [27]. The number of robots in the team is n_r. The number of nodes is the number of robots plus one ($n_{nodes} = n_r + 1$), because one should consider the goal as a node since the main objective is to shoot the ball to the goal. The graph nodes are the team robots, and the connections are all the possible actions with different weights based on the probability of the action, either passing between two robots or scoring a goal between a robot node and the goal node.

$$n_{con} = \frac{n_{nodes} * (n_{nodes} - 1)}{2} = \frac{(n_r - 1) * n_r}{2} \tag{5}$$

The calculation of the shortest distance (highest probability path) is based on the pathfinder algorithm A* [28,29]. The input to the algorithm is the starting and ending nodes, and these are always the robot that has the ball and the goal node, respectively. The output is the order of nodes that make the highest probability path. The A* algorithm applied to this strategy outputs the highest probability path of scoring a goal given the current game situation and ball handler. A* algorithm runs at every planning iteration and outputs the set of actions that maximizes the probability of scoring a goal. When the overall probability is low, the kicking skill is delayed as much as possible in order to give the team more opportunities to find a better set of actions.

An example of a complete graph in a game situation is visible in Figure 9. The graph representation uses the probability values and not the arbitrary ones calculated by the log probability theorem so that it is easier for human visualization and interpretation purposes.

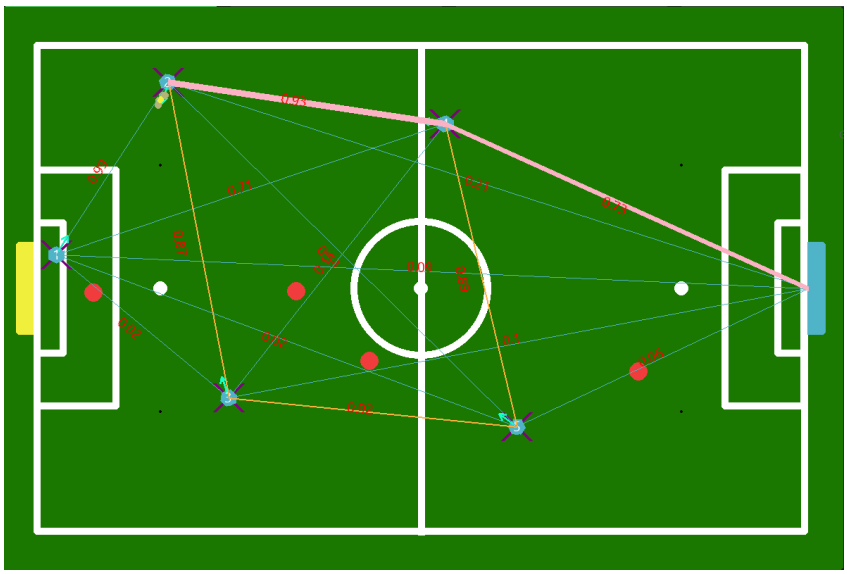


Figure 9. Complete graph in a game situation.

The highest probability path is represented in pink lines, and the probability of every action is also displayed near every line. To give robots more independence, besides the highest probability path it is also calculated the highest probability alternative. This is achieved by changing the arbitrary value of the first action to the highest value possible and then recalculating the highest probability path. The alternative is represented in orange, and both are given to the robot who has the ball. This is used for situations when the game changes too quickly and something prevents the robot with ball possession from making a pass. In those situations, the robot already knows the alternative to keep the strategy going. Since this is calculated at every loop, nothing needs to be performed if the robot decides to use the alternative path instead of the main one.

6. Decision Trees

Probability-based plays cover the path desired for the ball in offensive situations, but something needs to decide what type of situation the game and each of the robots are in at every moment. This is where decision trees come into place, and for this strategy, there are two of them [19,20].

6.1. Game Decision Tree

The game decision tree handles the game state and decides the skills desired for the situation represented in Figure 10. Each possible path in the game decision tree outputs a list of five skills. These skills are distributed between the players based on the other players' decision trees (explained further). The tree can be readjusted and configured by a JSON file.

Most of the branches have two possible outcomes that are solvable with a Boolean variable, for example, the first one is ***Ball***. The negative of that is ***not Ball*** or ***!Ball***. All the variable meanings are available in Table 3.

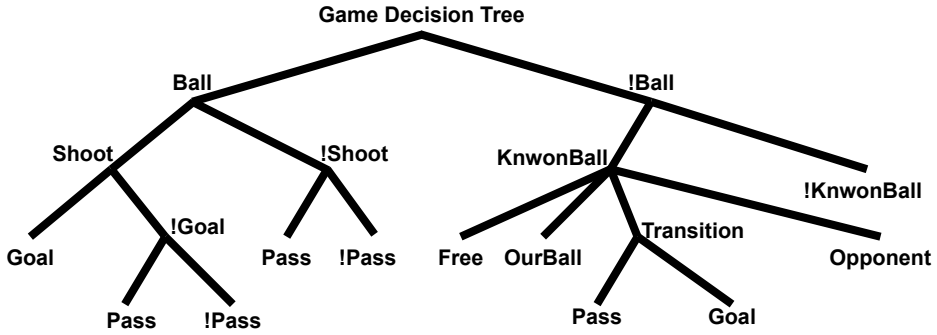


Figure 10. Game-state decision tree.

Table 3. Table of variables in the game decision tree.

Variable	Meaning
Ball	Team ball possession
Shoot	Team can perform a kick
Goal	Team is allowed by the rules to kick to the goal
Pass	Team can perform a pass
KnownBall	Team knows where the ball is
Free	The ball is free in the field with no nearby robot
OurBall	No ball possession but one of the teammates is the closest
Transition	The ball is in between an action, moving after a kick
Opponent	Opponent team has the ball or is the closest
Pass	The transition is a pass from our team
Goal	The transition is a goal kick

6.2. Player Decision Tree

The Player Decision Tree is calculated for every team member and decides which state that player is in, as shown in Figure 11. It is based on their current strategy, game and time states.

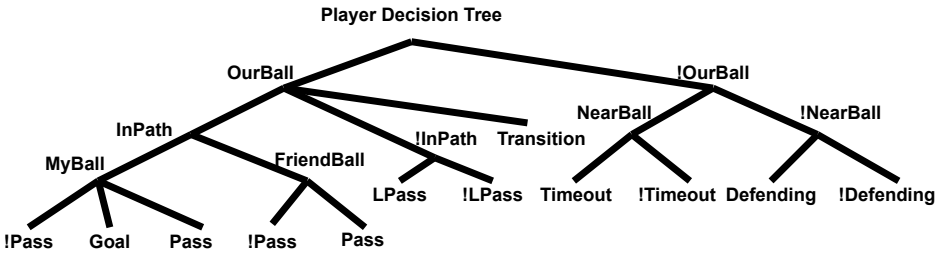


Figure 11. Player Decision Tree.

The same logic of variables in the game decision tree is applied here, for example, ***OurBall*** represents the ball possession and the ***!OurBall*** is the opposite. The Player Decision Tree is used to decide each robot's skill and even some arguments for that player state. Also associated with that player state are the weights of each of the heat maps that decide the robot's positioning. These weights are divided by player situation and not by robot, based on the fact that different situations are what determine different outcomes, and not the number of robots present on the field. Also, since the robots influence each other, even though they might be in the same player situation, their final heat maps are different, and so are their desired position. The Player Decision Tree variables are presented in Table 4.

Table 4. Table of variables in the Player Decision Tree.

Variable	Meaning
OurBall	Team ball possession
InPath	Robot is in the ideal ball path
Transition	The ball is between actions
MyBall	Robot has the ball
FriendBall	A teammate has the ball
Pass	A pass can be performed
Goal	The next action is a kick to the goal
LPass	There are lines of pass available
NearBall	Robot is the closest to the ball
Timeout	Timeout to attack the robot with the ball
Defending	Robot is defending an opponent

7. Heat Maps and Positioning

All the robot skills are based on decision trees and the Play Generator, but the move skill requires additional argument calculations. The calculations of the robot's positioning need to have multiple influences and possible solutions, and change according to the current game and player variables. Based on that, maps were created for all possible different situations that are as flexible as the game needs [21].

7.1. Information Used for Positioning

As predefined in the rules, all the information the teams can use has to be acquired by the team robots. Even though there is a lot of information about the robots themselves, it is used the location of all robots, the ball, and the game and player's state to position the robots accordingly.

7.2. Zones

In a human football team, each player has a different role and zone to play, with different possible combinations of zones. The zone formation varies based on the number of players on each team, and the RoboCup Middle Size League is comparable to five-a-side football because it also has five robots on each team. One of the most used zone formations is called "diamond". This formation has a goalkeeper, one defensive player, two mid-field players, and a striker in the front [30], visible in Figure 12.

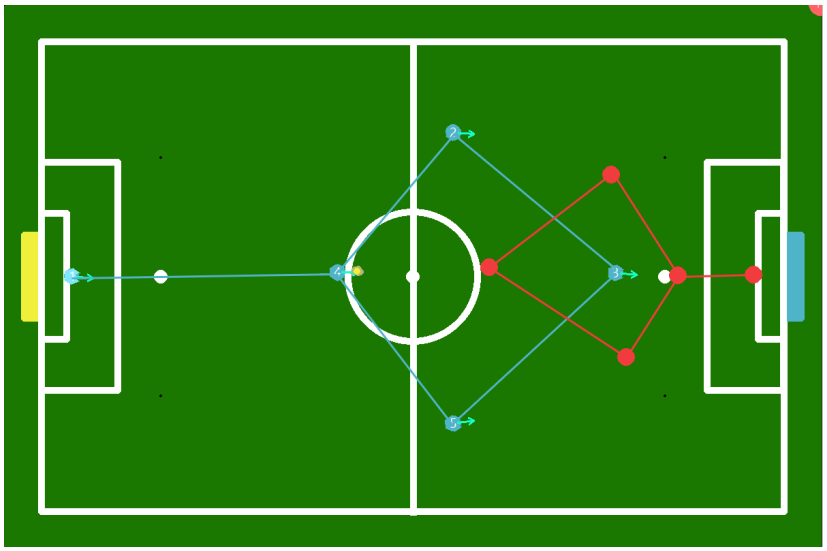


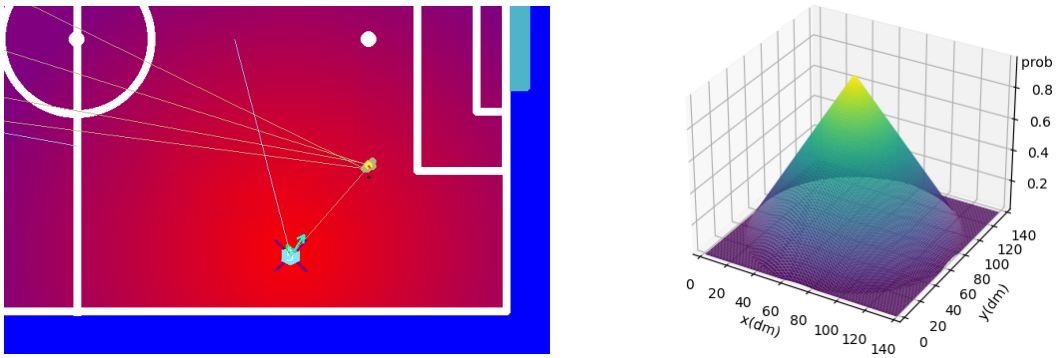
Figure 12. Offensive (blue) and defensive (red) diamond formation configurations.

The difference when it comes to applying these zones to football robots is that robots have the advantage of having the same physical skills, so there is no need to lock them in a designated zone. Changing zones is essential to take advantage of the current player's position and always to make sure that the furthest-back player, the least important in an offensive play, stays in the back as a safeguard [30]. The simplest way to assign a zone can be by the nearest player to that zone. However, that is not enough, because if a player is really off the zone, one player can be the closest to multiple zones, which is something undesirable. To apply this principle and to obtain a method to avoid this situation, the approach used was the Hungarian Algorithm [31] based on the distance of all the robots to their zone central points.

The Hungarian Algorithm uses matrix calculations to organize the robot's location according to the distance to each zone. The nearest zone is assigned to each robot in a way that each robot gets one and only one zone. This algorithm is calculated on each strategy cycle, even though the zone influence is not used in every situation.

7.3. Maps Generated

Each robot has a positioning map, which is the fusion of multiple different base maps. The base maps are pre-generated matrices of weights that represent a specific influence and have a formula based on the outcome desired by that influence [21]. A simple example is the distance base map, which is generated based on the distance from the robot's current position, visible in Figure 13. This map is designed to constrain player movement, effectively eliminating options that are excessively distant from the robot's current position.



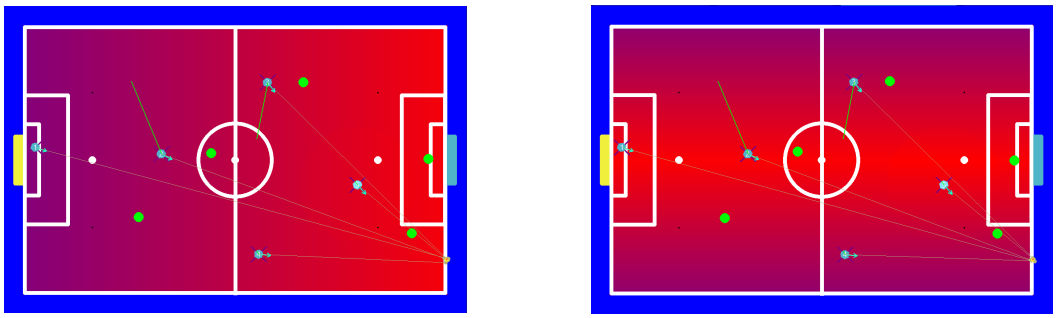
(**a**) (**b**)

Figure 13. Representation of a base map based on the distance to the robot: (**a**) Two-dimensional base station representation. (**b**) Three-dimensional representation.

This example is a linear equation based on distance, but other formulae have been used to generate different types of maps. Some of the most important are shown in a total of thirteen different maps. The software developed is modular and is able to accept as many maps as needed.

7.3.1. Middle Field and Center Field

Figure 14 shows two base maps used for a similar purpose with different influences. The first (a) influences to make the robots move forward in the field, and the second (b) is to make the robots avoid the sidelines. The second map has the purpose of making the robots play more in the middle of the field in order to avoid losing the ball by the sideline [30].

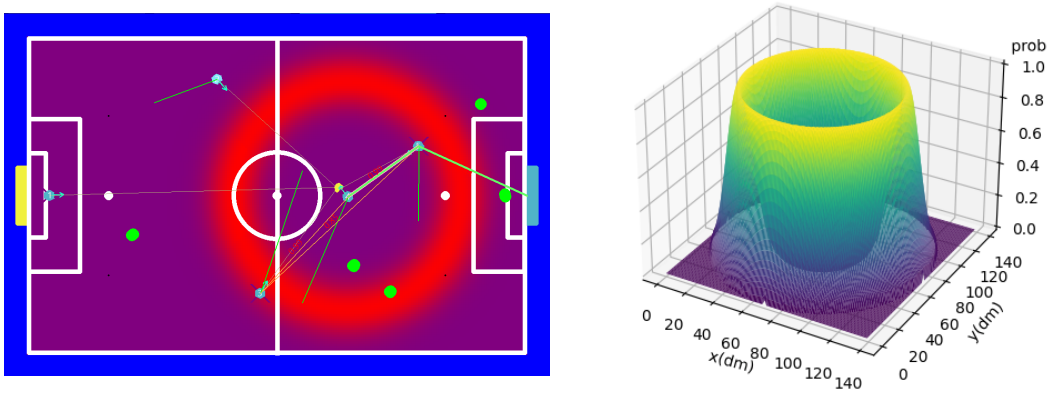


(**a**) (**b**)

Figure 14. Representation of base maps: (**a**) Middle field. (**b**) Center field.

7.3.2. Ideal Pass Distance

One way to optimize the overall probability is to position the robots in ideal conditions to pass. This base map was created for that purpose and varies from robot to robot depending on where the ideal pass distance map is positioned. This map is visible in Figure 15, where (a) is a field representation, and (b) is the Bayesian curve in a 3D representation [32]. For the robot in possession of the ball, the center of this matrix aligns with the next robot on the path. Conversely, for the robots positioned on the path but not in possession of the ball, the matrix's center is aligned with the robot from which they are anticipated to receive the ball. The formula employed for these calculations is identical to the one used for the ideal pass curve, which can be adapted for real-world applications.



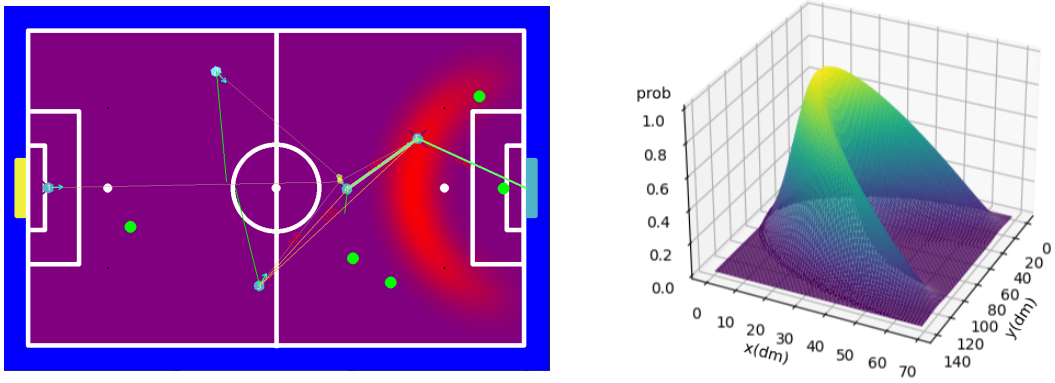
(**a**) (**b**)

Figure 15. Ideal pass distance base maps: (**a**) Field representation. (**b**) Three-dimensional pass probability representation.

7.3.3. Ideal Goal Distance

The same principle of the ideal pass distance is applied to the ideal goal distance but with some changes due to the importance of the kick angle to the goal. When performing a pass, both robots are facing each other, but when kicking to the goal, the angle from where one shoots influences the probability of scoring a goal. The formula for this matrix is the equation for the ideal goal distance, a Bayesian curve, times the cosine of the angle α, Equation (6). Visible in Figure 16 is the matrix applied to the field (a) and the 3D representation of the matrix (b). Besides the same variables as presented in the ideal pass distance formula, $dist_{RG}$ represents the distance between a robot and the goal, and IGD is the ideal goal distance.

$$GoalProb = cos(\alpha) * \frac{p}{d\sqrt{2\pi}} e^{\frac{-(dist_{RG}-IGD)^2}{2d^2}} \quad (6)$$

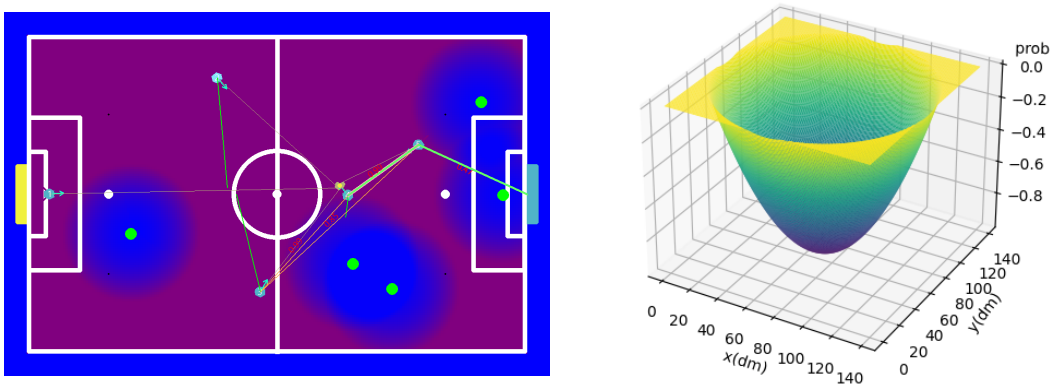


(**a**) (**b**)

Figure 16. Ideal goal distance base maps: (**a**) Field representation. (**b**) Three-dimensional probability representation.

7.3.4. Opponents Circle

In an offensive situation it is not ideal to be near the opponents. This map is a simple negative power function to make the robots avoid the opponents [21]. The matrix is the same for every opponent, and for optimization reasons, a map is created with all the opponent's matrices. The full map is the combination of n matrices, n being the number of opponents detected. Figure 17a represents the field with all the opponent's matrices, and Figure 17b is a 3D representation of an individual matrix.

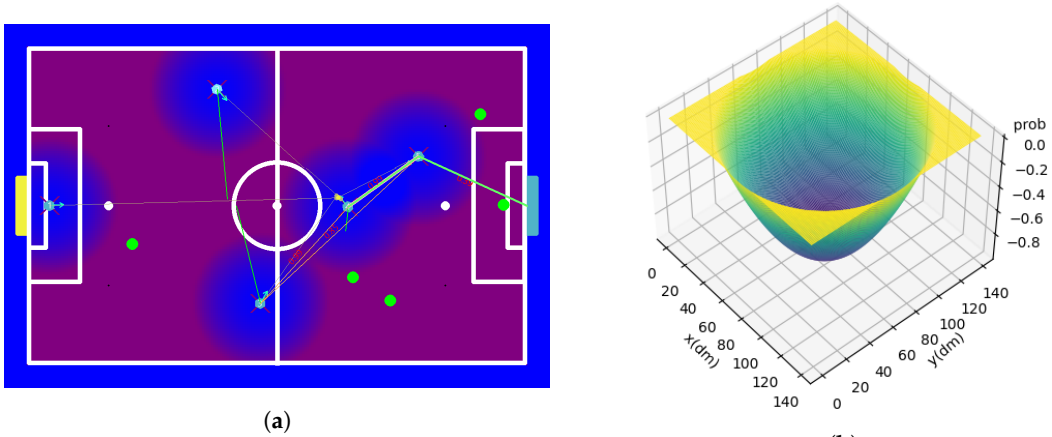


(**a**) (**b**)

Figure 17. Opponents influence: (**a**) Field situation. (**b**) Individual matrix.

7.3.5. Teammates

Just like in the opponent's situation, there is no advantage to having all the teammates near each other because that would be difficult for them to progress on the field. Also, if they try to avoid their teammates when they are in the same game situation, this map solves the problem of when both teammates try to get to the same position. In that situation, that position will be occupied by the first teammate to arrive there. Just like the opponent's matrix and map, the field map is generated for all teammates. Visible in Figure 18 is a field map (a) and the matrix used for every teammate (b).



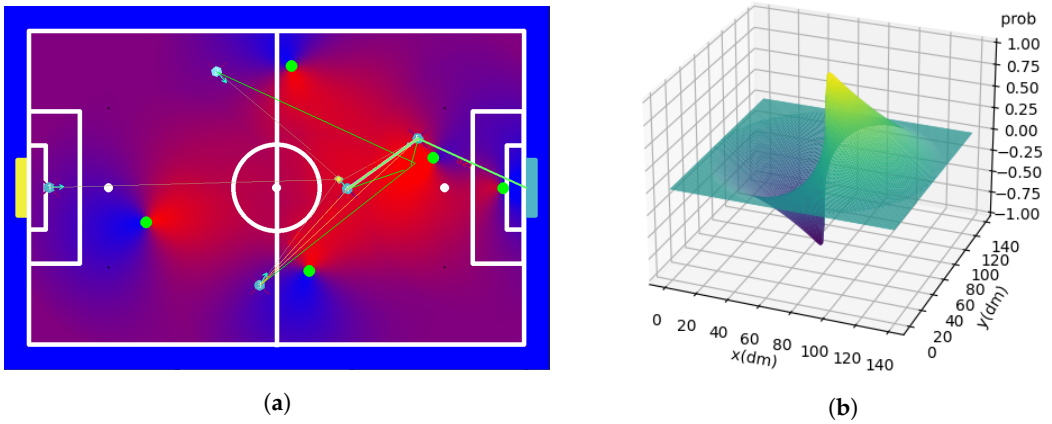
(**a**) (**b**)

Figure 18. Teammates' influence: (**a**) Field situation. (**b**) Individual matrix.

7.3.6. Opponents Cone

Even though opponents are avoidable in offensive situations, in defensive situations, the best way to position a robot is between an opponent and the ball. This covers that opponent and keeps the robot with the ball in sight. This influence is then created by the matrix presented in Figure 19b [30]. For optimization reasons and to achieve the resolution of one degree, 360 matrices were previously calculated. The angle between an opponent and the ball defines the angle of the matrix. This map is also generated with all the opponents present and used for all teammates, visible in Figure 19a. The formula used for the matrix is the cosine of the angle between the opponent and the ball multiplied by the distance to the opponent itself. In Equation (7), α is the angle between the opponent and the surrounding points, x_o and y_o are the opponent coordinates, x_1 and y_1 are the coordinates of the points around it, and r is the maximum radius affected by the field.

$$prob = cos(\alpha) * (\sqrt{(x_o - x_1)^2 + (y_o - y_1)^2} - r) \quad (7)$$

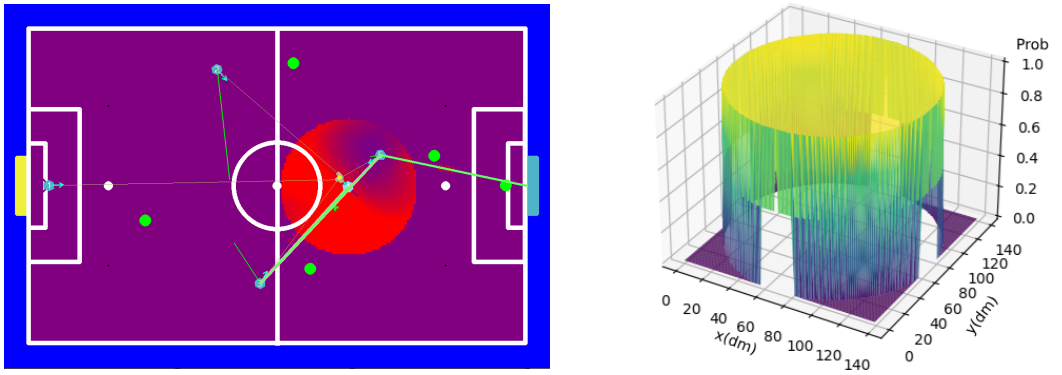


(**a**) (**b**)

Figure 19. Opponents Cone map: (**a**) Two-dimensional base station representation. (**b**) Three-dimensional representation.

7.3.7. Three-Meter Radius with the Ball

In order to make the games more appealing, there is a rule that limits the amount of space a robot can move with the ball (3 m radius). The rule is applied to the strategy using a map that limits how much a robot can move [18]. The matrix of this representation is presented in Figure 20b, where all the values inside the 3 m radius are 1 and all the other ones are 0. This field is generated for the robot that has the ball, and instead of summing to the other maps, this is multiplied by the outcome of all the other fields. A game situation of a player who has the ball is available in Figure 20a.



(**a**) (**b**)

Figure 20. Three-meter radius rule map: (**a**) Field situation. (**b**) Three-dimensional matrix representation.

7.4. Map Fusion

The position calculation is carried out every iteration in all robots, fusing all the maps. Each map has associated with it a ω value that is adjusted for different game situations. Every value of the matrix is multiplied by the ω weight. With all the matrices being the same size, its fusion can be achieved by summing all the maps, as shown in Equation (8).

$$M = \sum_{i=0}^{n-1} A_i \omega_i = \begin{bmatrix} \sum_{i=0}^{n-1} a_i(0,0)\omega_i & \sum_{i=0}^{n-1} a_i(1,0)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,0)\omega_i \\ \sum_{i=0}^{n-1} a_i(0,1)\omega_i & \sum_{i=0}^{n-1} a_i(1,1)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,1)\omega_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{n-1} a_i(0,y)\omega_i & \sum_{i=0}^{n-1} a_i(1,y)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,y)\omega_i \end{bmatrix} \quad (8)$$

The variable $a_i(x,y)$ represents the multiple base map matrices, $A_1, A_2, \ldots, A_n$ respectively, and ω_i is the associated weight of each map influence.

After calculating the matrix M, if the robot has the ball, the three-meter radius matrix is multiplied in order to limit the robot's movement. The heat map calculated does not return a specific position but a general area of where the robot should go, and based on that, the position can be calculated. A new binary matrix is created where the maximum values of the heat map are equal to 1 and the rest is 0. This binary map is then used to detect all the good positioning areas available. Within *OpenCV*, with the functions *cv2.findContours()* and *cv2.moments()*, the center of mass is calculated for every area. When multiple ideal areas are available, the selection of the best one is based on the largest area.

Figure 21 shows a situation of 4 maximum value areas, with their specific maximum value center of mass with a green cross, and its binary map. The location chosen is the center of mass of the largest area region, in this case, region B.

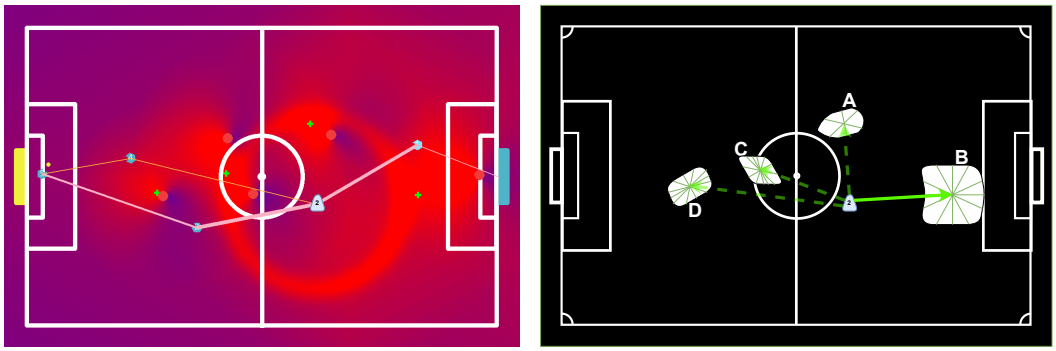
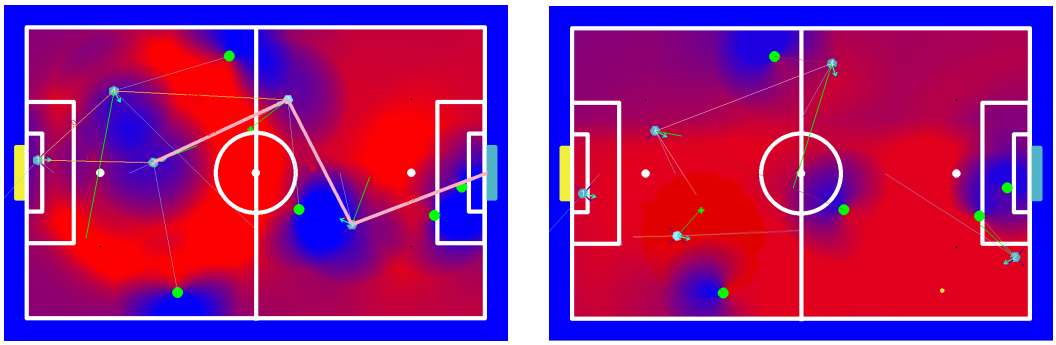


Figure 21. Position selection based on multiple possible areas.

Figure 22 shows two probability heat maps, one offensive and one defensive. The first map (a) is an offensive situation, where robot number 5 will receive a pass from robot 2. Its main objective is to get close to its teammate in order to increase the pass success

probability. The green lines represent the movement intention of every robot. Robot 3 will move to the center of the field to avoid staying behind the opponent and to get a clean line of pass from robot 5. The second map (b) shows a defensive situation where the robots show the intention to move to positions to cover one or more opponents. The map shows robot number 5's intention, where the zone and distance are the elements with more influence.



(**a**) (**b**)

Figure 22. Game probability maps: (**a**) Offensive situation. (**b**) Defensive situation.

7.5. Calibration

The influences of all maps have a certain weight associated with them, and these need to be adjusted according to the desired output of a certain game and player situation. For an output of the decision trees, the weight of the maps desired is calibrated so that the robots position themselves correctly. Each map can have an influence between 0 and 1, where 0 is no influence at all, and 1 is the maximum influence. These maps were calibrated using the simulator and based on normal football player behaviors in those situations.

With the sum of all heat maps visible on the screen, the simulator is used to emulate a play. In each game situation, the human coach stops the virtual game and adjusts each map contribution to values that feature the intended result. Then, the virtual play resumes, and this procedure is repeated until the robot's behavior is as intended. Using the simulator to calibrate the maps simplifies and accelerates the process because of the ability to create custom game situations to be studied. The calibration user interface is shown in Figure 23.

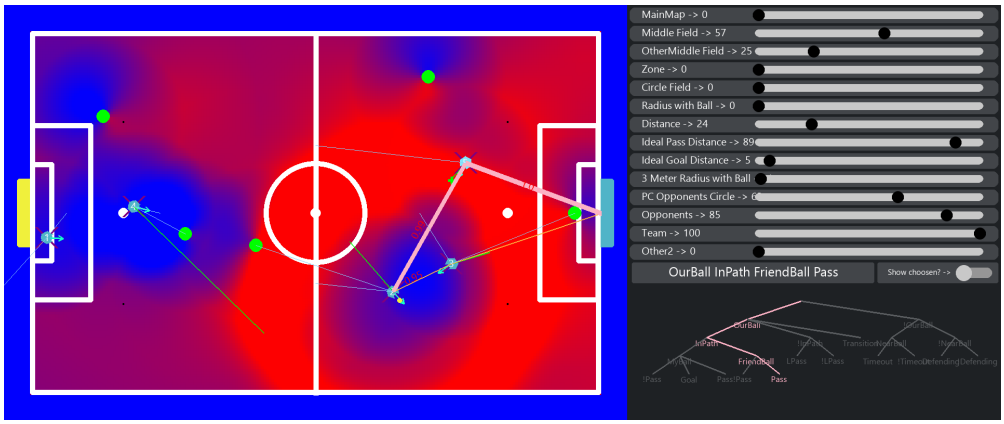


Figure 23. User interface for heat positioning map calibration.

8. Simulator

As an objective, LAR@MSL wanted to be able to play in less than a year, and so the high-level/low-level code and hardware needed to be developed in parallel. In order to develop the strategy, communication, and some logic, a Webots simulation environment was developed [33]. The choice of Webots was due to the open-source availability, high

performance, multiple platforms, and multiple methods of communication between the code and the robots in the simulation environment [34].

8.1. Architecture

Figure 24 shows a base architecture schematic of the simulation environment developed. Each robot present in the Webots world has its own controller [35]. A controller is the code that runs on each robot, handles the communication and information gathering, and controls the robot's movement and actions. From the base station, in a manner similar to the real world, the robot controllers receive skills along with their corresponding arguments and react accordingly.

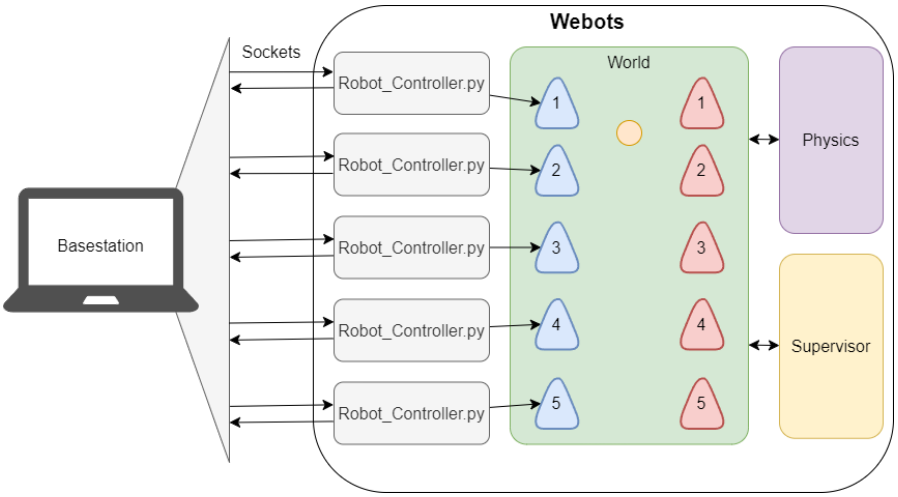


Figure 24. The architecture of the developed Webots simulation environment.

8.2. Simulator Elements

The simulation world has a field, a ball, and ten robots, five from each team. The robots are able to perform the essential parts for the strategy to be developed, implemented, and studied, but they are simple so that the computational requirements are not high.

8.2.1. Field, Goals, and Ball

The field and goals are solid-state elements present in the world simulation, and even though these entities are not affected by gravity, collisions are detected so that when some other object goes against it, they do not pass through. The ball, on the other hand, besides having gravity applied to it, also has an associated weight and elasticity so that the ball's behavior is similar to the real world. The dribbling system is simulated with controllable magnets in the robot and the ball. The ball has a passive magnet, and the robots can turn on or off their magnet in order to handle the ball.

8.2.2. Robot

Each robot in the simulation has an extruded hexagon shape and a magnetic actuator in the front. The robot has no wheels because there is no need to waste computational power on Omni wheels, PID control systems, or physics. The robot is controlled by applying a velocity, and the arguments are the same as used in the real world, such as linear velocity, direction, and angular Velocity. Besides the movement, the robot handles the ball dribbling mechanism and the kicking mechanism. The dribbling mechanism is based on magnetic actuators. When the ball is near the front of the robot, the magnetic actuator is activated and the ball is kept in the robot until the robot kicks or releases the ball. The kicking mechanism is implemented by code and not a physical actuator simulation. When the robot performs a kick, a force is applied to the ball, directing it along the orientation the robot is facing.

8.2.3. Opponent

The opponent robots are defined as extruded hexagon shapes on the field and are moved on the simulator. For reasons of ease of development, they are equal to the team robots to be easily controlled by the control skill. This approach allows testing of both sides of the strategy by putting teams with identical strategies against each other. This enables finding and fixing any issues or flaws and even arranging games between five human players and the robots. The five humans play with game controllers, and each human controls one robot against the strategy developed.

8.3. Communications System

In order to keep the transition between the real world and the simulator simple, the communication used in the real world and in the simulator is the same and uses sockets [36]. The only change between the simulator and the real world is the robot's IP addresses, which in the simulator are all the same, 127.0.0.1.

8.3.1. IPs

As stipulated by the RoboCup Rules, each RoboCup MSL team has a specific network mask. The robot's IP ranges from 172.16.49.11 to 172.16.49.15, and the team mask for LAR@MSL is 24 bits (172.16.49.XXX) [18].

8.3.2. Web Sockets

Figure 25 shows the port organization between the base station, the real-world robots, and the simulator. Each robot has two socket communications and four ports, two per socket. Even though sockets are bidirectional, for performance and organization reasons, they only communicate one way. This was thought to be due to the buffers, the handlers, and the frequency in which the data flow from one side to the other, which have different rates and sizes.

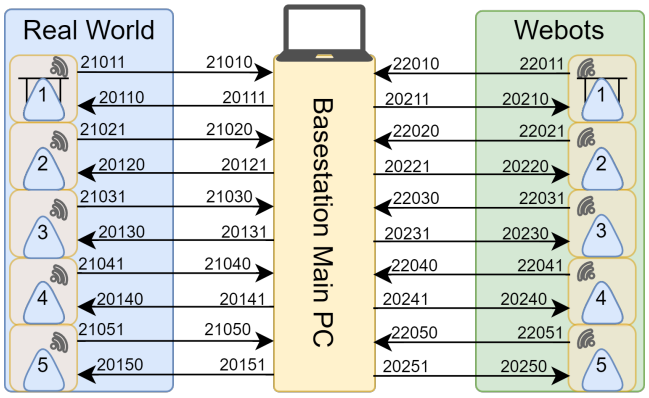


Figure 25. Communications organization in real-world and simulator.

8.4. Real-World Required Adaptations

The formulae used in the simulator need adjustments when the transition to the real world is carried out. There are two types of adjustments: value adjustments and game adjustments. Regarding the value adjustments, the changes made are in the probability of pass and goal formulae. Since these formulae depend on the robot control systems, after the implementation of these systems, tests were carried out to obtain the probability of passing at multiple distances. The same was carried out for the goal probability.

The opponent's influence, as well as some weights of the heat maps, falls under the category of game adjustment. The changes are adapted to different opponent teams, and with the same modular strategy, the robot's behavior is adjusted accordingly. For example, when playing against a team with fast robots, the opponent's influence on the lines of the pass is increased to force the robots to make safer passes, reducing the overall risk.

9. Results

The strategy had different study approaches: in the simulator on specific plays, in the simulator but against a human-controlled team, and in the real world. There were differences between the simulation and the real world (RoboCup games), as expected. All the data acquired in the real world were acquired in the RoboCup 2023 competition, where the LAR@MSL team was able to play with the best in the world. Even with the transition between the simulation and the real world, as well as the competition environment, the strategy was in play in all games during the competition, and the results were visible.

9.1. Simulator

As expected, the simulation behavior was almost flawless regarding the control systems and information, and that helped the development of the strategy itself. Besides controlling the robots from the team and simulating the communication system, opponents were also positioned to test different game situations.

9.1.1. Case 1

The first game situation, presented in Figure 26, is an offensive situation. After robot 2 recovers the ball, the highest probability path is from robot number 2 to robot number 4, and then to robot number 5 to shoot to the goal. While that happens, robot number 3 tries to position itself near the ideal goal distance to give an alternative to both robot numbers 4 and 5. If robot number 4, after receiving the ball, has no line of pass to 5, it already has number 3 as an alternative line of pass. This is an example of where the heat maps and probability path complement each other by trying to optimize each other.

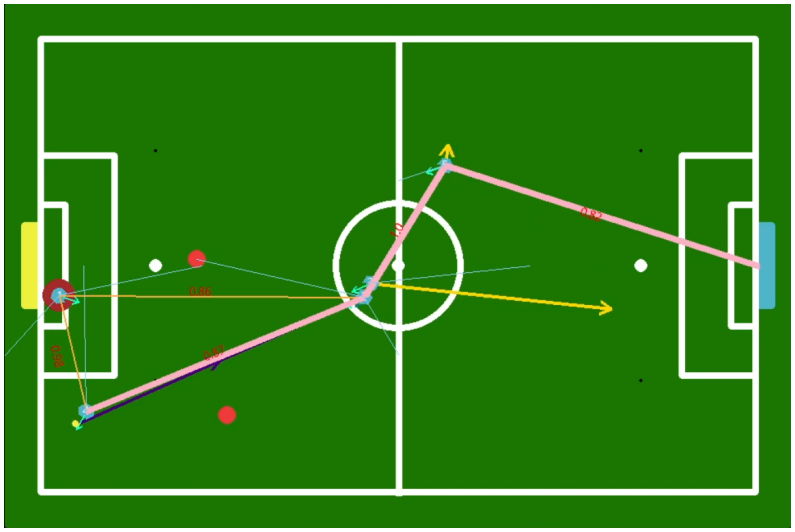
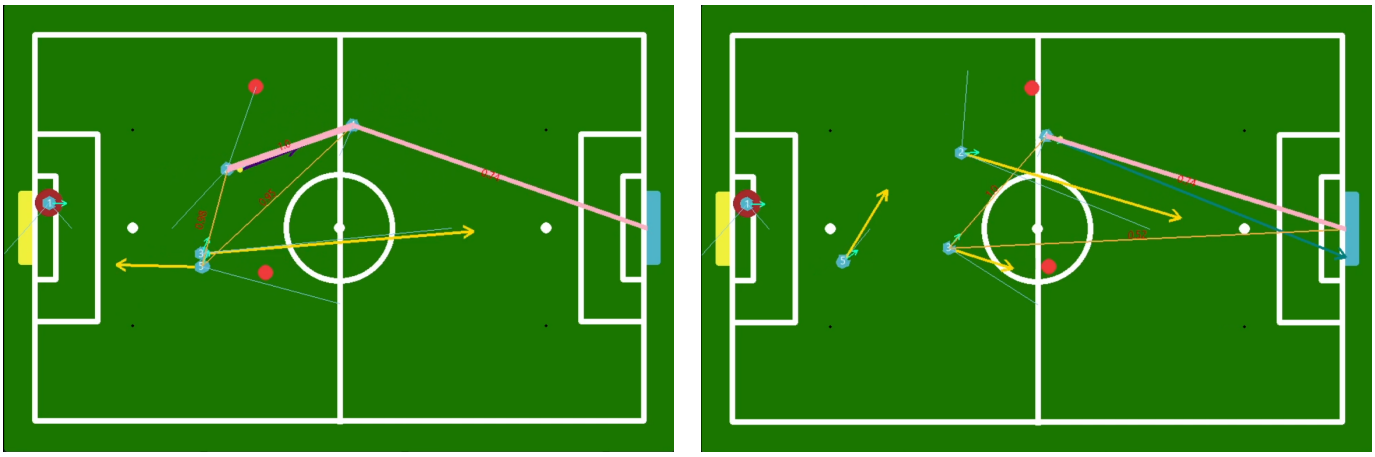


Figure 26. First game situation in the simulator.

9.1.2. Case 2

Figure 27 has two images that are from the same game scenario, but image (b) is the situation after the first pass. In Figure 27a, robot number 2 has the ball, and the highest probability path states that the following action is to pass the ball to robot number 4. At the same time, robot number 5 will stay further back, trying to defend, and robot number 3 will go forward, trying to position itself to give robot number 4 an alternative line of pass. After the first pass between robots number 2 and number 4 is made, robot number 5 positions itself more to the center, and robots number 3 and number 2 change tasks. This happened because robot number 2 was closer to the desired spot. Since there are no other opponents detected, robot number 4 also has a relatively high probability of scoring, that being the task assigned to it.



(**a**) (**b**)

Figure 27. Third game situation in the simulator: (**a**) Initial situation. (**b**) Situation after the first pass.

9.1.3. Case 3

Another offensive play is visible in Figure 28, where the probability of pass and goal is high, but the position is the key difference. Robot number 3 stays near its assigned zone and becomes a more defensive player, while robot number 5 tries to position itself between robot number 2, the one with the ball possession, and robot number 4. This intention is based on always trying to create alternative passes to maintain a fast game.

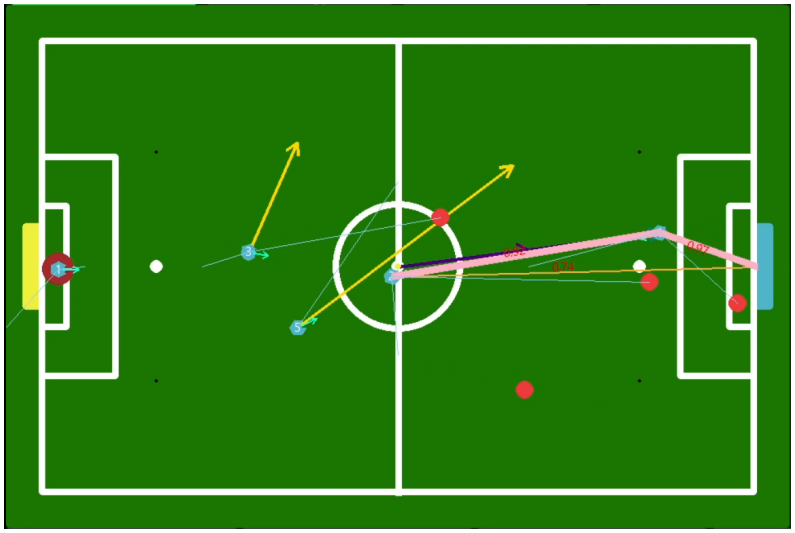


Figure 28. Second game situation in the simulator.

9.1.4. Case 4

Now, visible in Figure 29, is a defensive situation where each robot has an opponent assigned based on the Hungarian Algorithm, visible with the dark red lines. The robot of the assigned opponent must defend and try to cut the line of pass or goal. As a way to maintain high pressure, the robot closest to the ball still tries to attack it, in this case, it is robot number 2. Robot number 3, besides covering the opponent's line of goal further back, is deliberately covering that position, with the intent of also cutting the line of pass to the opponent near the center.

9.2. Against Humans

After completing the calibration, the tests needed to be more dynamic. As a result, developing a simulator that allows humans to play against robots became a need. Just like in common football computer games like FIFA, a game controller was given to each human

to control one of the opponent robots. Using the control skill, each human handled the robot's movement, orientation, and pass or kick actions. Playing against a team of humans was carried out specifically to help better understand what types of in-game situations needed some adjustments.

Some videos of the strategy development are available in Appendix A, where the simulator, the base station, and the strategy actions sent to the robots are shown.

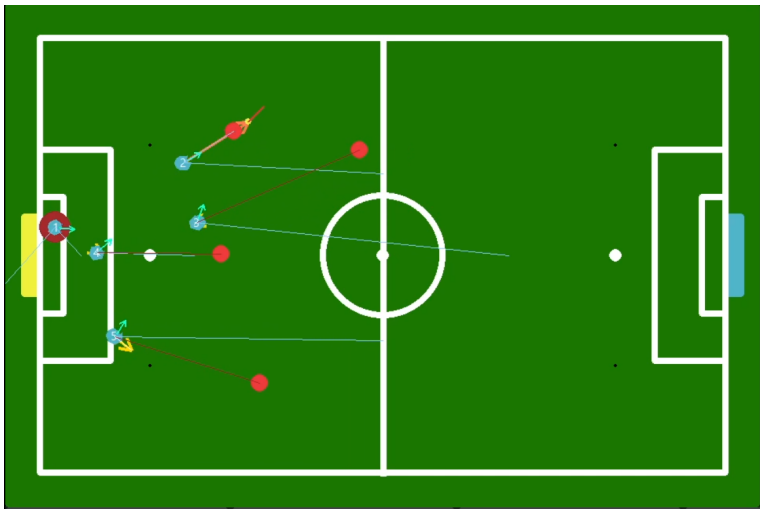


Figure 29. Forth game situation in the simulator.

9.3. Real World

The transition to the real world was studied on different occasions, but games were only played in RoboCup 2023. A short video that shows the intentions and defensive plays of some games played in the competition is available in Appendix B. It is visible in the real-world image (filmed by one of the competition teams) and the base station information gathered by the robots and the decisions taken. During the competition, all the games were recorded, and all the information sent to and from the robots was recorded into a log file. This video shows four different plays, which are described next.

9.3.1. Play 1

The play shows an offensive situation starting from kick-off. Immediately after kick-off, the players tried to position themselves to receive a pass and try to score a goal. This happened because they were on a more defensive and quick counter-attack strategy, defined by the maps and probability curves. It is visible that after kick-off, robot 3 tries to position itself in a line of pass and goal, but due to the poor robot control system, it was not able to receive the ball from robot 2.

9.3.2. Play 2

This shows a defensive situation, where all the robots were positioning themselves in a way to go to either side of the field. This opponent's kick was saved by the goalkeeper's good positioning, who managed to save the goal based on the information given by the other teammates.

9.3.3. Play 3

In play 3, a defensive situation, a synchronization problem caused by the saturated Wi-Fi environment in the competition is visible. However, the robots still behaved properly. Although robot number 4 did not move due to hardware problems, robot number 2 covered the lines of goal and pass. After the opponent's pass, the detection system malfunctioned, and the robot was not correctly positioned to defend the goal or cut the line of pass.

9.3.4. Play 4

This example shows a defensive corner situation where the team tries to defend the goal. Two of the robots were out of the game due to mechanical and hardware problems. With the three robots on the field, robot number 2 and robot number 3 try to cut lines of goal from multiple robots while robot number 1, the goalkeeper, tries to position itself to cover the ball. The goal lines are covered with the skill cover, and the position on the goal line of robot 3 was further back so that the goal area covered would be higher. Since they were at a numerical disadvantage, there was no point in robot number 3 covering the nearest opponent.

9.3.5. Real-World Differences

Due to the project's recency, the control system experienced performance issues when transitioning from simulation to the real world, consequently affecting the strategy. The information gathered by the robots was inconsistent, leading to less effective real-world results compared with their performance in simulations. The overall strategy output remained accurate, but the robots' data inaccuracies and simpler control systems limited the effectiveness of the strategy. Most of the effective plays were defensive, aimed at cutting lines of pass and goal. In this competition, precise robot control is crucial, especially as opposing teams had faster robots than those of the LAR@MSL team.

The communication reliability is also an important factor, although robots are prepared to play with some packet loss. Robots communicate with the base station at a 25 Hz frequency. For this strategy to work, it is advisable to have at least one packet per second. Should the loss be higher, the robots will continue playing but with a decrease in performance and stability. To counteract the packet loss effects, an alternative play is generated and given to the robots. Data were taken from multiple games during RoboCup, and the communications between the five robots and the base station had an average of 86% success packets. This proves the reliability of the UDP sockets used by the strategy in highly saturated environments.

10. STP vs. PBS

Since the strategy implemented was developed for the RoboCup championship, the comparison between STP and PBS is a good way to explain the advantages and disadvantages of the work developed.

10.1. Differences

Both strategy algorithms are based on skills that the robots are capable of performing, but the way these are attributed to the robots is very different. While the STP architecture is decentralized and the synchronism is maintained, the PBS is centralized and depends on a reliable and efficient communication system. Another major difference is the scalability of the architecture. STP decides which play to use at each moment based on the plays developed in the playbook. One of the problems is that the plays are thought to have a static number of players, and so if the number of players in a team changes, the playbook needs to be rebuilt. PBS skips the need to implement tactics, and that makes it simpler, easier, and less time-consuming to implement.

10.2. Limitations

PBS also has some disadvantages, and the limitations are almost all based on information, actions, and the opponent's behavior. Firstly, since the decision-making is centralized, the latency of communications can be a problem. Therefore, the communication system implemented needs to be as fast and reliable as possible. It is also very dependent on the information given by the robots, like their locations and the objects they detect in the field. If that information is not accurate and precise, it can easily change the output of the decisions taken. Regarding the control systems, if the robots do not behave well given a certain skill, the outcome of the game will be ruined even though the strategy still tries to

give the best output possible. The strategy is as efficient, as the robots are able to perform the skills fast and reliably.

Secondly, the first calibration process of the heat maps can be time-consuming. Even though the calibration is initially set up in the simulator and can be primarily tested against humans, it can take quite some time to achieve a stable solution. After the first stable solution, adapting and changing the behavior is quicker and more straightforward.

PBS also depends on the opponent's behavior, and the strategy results in the real world were as good as the opponent team's performance. If the opponent team has a stable, organized strategy, it is easily predictable and excelled by PBS. If the opponent team's behavior is entirely random, PBS makes decisions based on unexpected events and performs worse.

10.3. Modularity, Flexibility, and Scalability

The modularity, flexibility, and scalability of the strategy were all factors taken into consideration while designing the PBS algorithm.

Concerning the system modularity, the system is easy to change and adapt to different game situations and opponent's behavior. Simply by adjusting the heat map weights, as well as the opponent's influence in the pass and goal probability formulae, the risk taken by the robots can be adapted to the opponent team.

Regarding flexibility, both the game and Player Decision Trees are easily configurable to adapt to any opponent team strategy.

PBS is easily scalable since it does not rely on the number of players on either team.

11. Conclusions

A new method of decision-making regarding the strategy of multi-agent autonomous robots can greatly improve football strategies and robotics in general. The strategy architecture presented solves a robotics problem without a typical human solution but with a mathematical and robotics approach.

The strategy proved modular, flexible, and easily adaptable to different game situations with minimal effort. It allows different behaviors against different opponents and can support any team size, either its own or the opponent's team.

Besides being extremely customizable, it requires less time to implement the strategy when compared with STP, and the time required to adjust any parameter is also negligible. Another important advantage is that this method eliminates the need for plays and a playbook, which are very time-consuming.

Simulating the environment was also a key factor in accelerating the development, testing, and even calibrating the strategy and probability formulae. Controlling the opponents in the simulation was also an important tool that helped better understand how the strategy behaves in certain situations, making it able to test in different modes, such as the following:

- Manually positioning an opponent;
- Strategy vs. manually controlled by the base station;
- Strategy vs. humans controlling the opponents with gamepads.

The work developed here relies on a reasonably reliable communication system. In some saturated environments, the reliability of the network can become compromised, but the stability of the communication system developed proved to be stable.

Author Contributions: Conceptualization, A.F.A.R., N.S.S.M.P. and G.T.L.; methodology, A.F.A.R., A.C.C.L. and T.A.R.; software, A.F.A.R., A.C.C.L. and A.F.M.R.; validation, A.F.A.R., A.C.C.L., G.T.L. and A.F.M.R.; formal analysis, A.F.A.R., T.A.R. and N.S.S.M.P.; investigation, A.F.A.R., A.C.C.L., T.A.R., N.S.S.M.P., G.T.L. and A.F.M.R.; resources, N.S.S.M.P., G.T.L. and A.F.M.R.; data curation, A.F.A.R., A.C.C.L. and T.A.R.; writing—original draft, A.F.A.R.; writing—review and editing, A.C.C.L., T.A.R., N.S.S.M.P., G.T.L. and A.F.M.R.; visualization, A.F.A.R., G.T.L. and A.F.M.R.; supervision, T.A.R., N.S.S.M.P., G.T.L. and A.F.M.R.; project administration, G.T.L. and A.F.M.R.; funding acquisition, G.T.L. and A.F.M.R. All authors have read and agreed to the published version of the manuscript.

Funding: The third author received funding through a doctoral scholarship from the Portuguese Foundation for Science and Technology (Fundação para a Ciência e a Tecnologia) [grant number SFRH/BD/06944/2020], with funds from the Portuguese Ministry of Science, Technology and Higher Education and the European Social Fund through the Programa Operacional do Capital Humano (POCH). This work has been supported by FCT—Fundação para a Ciência e a Tecnologia within the R& D Units Project Scope: UIDB/00319/2020.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work has been supported by the Laboratory of Automation and Robotics (LAR) of the University of Minho and the ALGORITMI research center.

Conflicts of Interest: Author Nino Sancho Sampaio Martins Pereira was employed by the company Dyson Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. "LAR@MSL Strategy: Humans vs. Robots in the Simulator"

https://www.youtube.com/watch?v=SVYgXKeGaf0.

Appendix B. "LAR@MSL Strategy: RoboCup Plays"

https://www.youtube.com/watch?v=63nJXTCfQEk.

References

1. Nardi, D.; Noda, I.; Ribeiro, F.; Stone, P.; von Stryk, O.; Veloso, M. RoboCup Soccer Leagues. *AI Mag.* **2014**, *35*, 77–85. [CrossRef]
2. Ribeiro, A.; Costa, J.; Martins, J.; Silva, R.; Lima, R.; Lopes, C.; Lopes, G.; Ribeiro, A.F. LAR@MSL Description Paper 2023. Available online: https://lar.dei.uminho.pt/images/downloads/LAR@MSL_TDP%202023.pdf (accessed on 13 October 2023).
3. Stone, P. Will Robots Triumph Over World Cup Winners by 2050? Available online: https://spectrum.ieee.org/robocup-robot-soccer (accessed on 28 October 2023).
4. Mendoza, J.P.; Biswas, J.; Zhu, D.; Wang, R.; Cooksey, P.; Klee, S.; Veloso, M. CMDragons 2015: Coordinated Offense and Defense of the SSL Champions. In *RoboCup 2015: Robot World Cup XIX. RoboCup 2015*; Almeida, L., Ji, J., Steinbauer, G., Luke, S., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 9513. [CrossRef]
5. Biswas, J.; Mendoza, J.P.; Zhu, D.; Choi, B.; Klee, S.; Veloso, M. Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In Proceedings of the AAMAS '14: International conference on Autonomous Agents and Multi-Agent Systems, Paris, France, 5–9 May 2014; Volume 1, pp. 493–500.
6. Browning, B.; Bruce, J.; Bowling, M.; Veloso, M. STP: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proc. Inst. Mech. Eng. Part J. Syst. Control Eng.* **2005**, *219*, 33–52. [CrossRef]
7. de Koning, L.; Mendoza, J.P.; Veloso, M.; van de Molengraft, R. Skills, Tactics and Plays for Distributed Multi-robot Control in Adversarial Environments. In *RoboCup 2017: Robot World Cup XXI. RoboCup 2017*; Akiyama, H., Obst, O., Sammut, C., Tonidandel, F., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 11175. [CrossRef]
8. Schwab, D.; Zhu, Y.; Veloso, M. Learning Skills for Small Size League RoboCup. In *RoboCup 2018: Robot World Cup XXII. RoboCup 2018*; Holz, D., Genter, K., Saad, M., von Stryk, O., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11374. [CrossRef]
9. Orr, J.; Dutta, A. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* **2023**, *23*, 3625. [CrossRef]
10. Antonioni, E.; Suriani, V.; Riccio, F.; Nardi, D. Game Strategies for Physical Robot Soccer Players: A Survey. *IEEE Trans. Games* **2021**, *13*, 342–357. [CrossRef]
11. Wu, J.; Snášel, V.; Cui, G. A Graph Theory-Based Evaluation of Strategy Set in Robot Soccer. In *Intelligent Data Analysis and Its Applications, Volume I. Advances in Intelligent Systems and Computing*; Pan, J.S., Snasel, V., Corchado, E., Abraham, A., Wang, S.L., Eds.; Springer: Cham, Switzerland, 2014; Volume 297. [CrossRef]
12. Caputo, R.R.; Santos, E.B.d. Bayesian Classifiers Supported by Ranking for Decision Making in Robot Soccer. In Proceedings of the 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), Sao Paulo, Brazil, 22–25 October 2018; pp. 432–437. [CrossRef]
13. Budanov, D.; Feltracco, J.; Kamat, J.; Medrano, R.; Naeem, S.; Neiger, J.; Osawa, R.; Pan, M.; Peterson, E.; Shaw, A.; et al. RoboJackets 2017 Team Description Paper. Available online: https://ssl.robocup.org/wp-content/uploads/2019/01/2017_TDP_RoboJackets.pdf (accessed on 18 December 2023).
14. Abe, T.; Orihara, R.; Sei, Y.; Tahara, Y.; Ohsuga, A. Acquisition of Cooperative Behavior in a Soccer Task Using Reward Shaping. In Proceedings of the 2021 5th International Conference on Innovation in Artificial Intelligence (ICIAI '21), Xiamen, China, 5–8 March 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 145–150. [CrossRef]

15. Liu, S.; Lever, G.; Wang, Z.; Merel, J.; Eslami, S.M.A.; Hennes, D.; Czarnecki, W.M.; Tassa, Y.; Omidshafiei, S.; Abdolmaleki, A.; et al. From Motor Control to Team Play in Simulated Humanoid Football. Available online: https://arxiv.org/abs/2105.12196 (accessed on 23 October 2023).
16. Tavafi, A.; Banzhaf, W. A hybrid genetic programming decision making system for RoboCup soccer simulation. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17), Berlin, Germany, 15–19 July 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1025–1032. [CrossRef]
17. Lima, P.; Bonarini, A.; Machado, C.; Marchese, F.M.; Marques, C.; Ribeiro, F.; Sorrenti, D.G. Omni-Directional Catadioptric Vision for Soccer Robots. *Robot. Auton. Syst. J.* **2021**, *36*, 87–102. Available online: https://hdl.handle.net/1822/3173 (accessed on 21 October 2023). [CrossRef]
18. Middle Size Robot League Rules and Regulations for 2023. Available online: https://msl.robocup.org/wp-content/uploads/2023/06/Rulebook_MSL2023_v24.2.pdf (accessed on 21 October 2023).
19. Quinlan, J.R. Induction of Decision Trees. Available online: https://hunch.net/~coms-4771/quinlan.pdf (accessed on 13 October 2023).
20. Song, Y.Y.; Lu, Y. Decision tree methods: Applications for classification and prediction. *Shanghai Arch. Psychiatry* **2015**, *27*, 130–135. [CrossRef] [PubMed]
21. Wang, Y.; Shenhan, J.; Chen, Z.; Huang, Z.; Xiong, R. Multi-Agent Collaboration for Feasible Collaborative Behavior Construction and Evaluation. 2019. Available online: https://www.researchgate.net/publication/336146890_Multi-agent_Collaboration_for_Feasible_Collaborative_Behavior_Construction_and_Evaluation (accessed on 28 October 2023).
22. Santos, F.; Almeida, L.; Lopes, L.S.; Azevedo, J.L.; Cunha, M.B. Communicating among robots in the RoboCup Middle-Size League. Available online: https://sweet.ua.pt/lsl/pubs/CLI-2010-b-robocup09-final.pdf (accessed on 16 October 2023).
23. Olthuis, J.J.; Beumer, R.M.; Bogaert, R.v.; Hameeteman, D.M.J.; de Loo, H.C.T.v.; G, M.J.; Molengraft, V.d.; Teurlings, P.; Verhees, E.D.T. Available online: https://msl.robocup.org/wp-content/uploads/2022/12/Risk_Evaluation_of_Robot_Soccer_with_Humans_in_MSL.pdf (accessed on 8 October 2023).
24. O'Hagan, A. Bayesian Statistics: Principles and Benefits. Available online: https://edepot.wur.nl/134085 (accessed on 7 September 2023).
25. Log Probabilities. Available online: https://chrispiech.github.io/probabilityForComputerScientists/en/part1/log_probabilities/ (accessed on 8 September 2023).
26. Kass, R.E.; Vos, P.W. *Geometrical Foundations of Asymptotic Inference*; John Wiley & Sons: New York, NY, USA, 1997; p. 14, ISBN 0-471-82668-5. Available online: https://www.wiley.com/en-us/Geometrical+Foundations+of+Asymptotic+Inference-p-9780471826682 (accessed on 28 October 2023).
27. Gowda, D.V.; Shashidhara, K.S.; Ramesha, M.; Sridhara, S.B.; Kumar, S.B.M. Recent Advances in Graph Theory And Its Applications. *Adv. Math. Sci. J.* **2021**, *10*, 1407–1412. [CrossRef]
28. Ribeiro, F.; Moutinho, I.; Pereira, N.; Oliveira, F.; Fernandes, J.; Peixoto, N.; Salgado, A. High accuracy navigation in unknown environment using adaptive control. In Proceedings of the RoboCup 2007: Robot Soccer World Cup XI, Atlanta, GA, USA, 9–10 July 2007; Lecture Notes on Artificial Intelligence Series; Springer: Berlin/Heidelberg, Germany, 2017; Volume 5001, pp. 312–319. [CrossRef]
29. Pereira, N.; Ribeiro, A.F.; Lopes, G.; Whitney, D.; Lino, J. Path planning towards non-compulsory multiple targets using TWIN-RRT*. *Ind. Robot. Int. J.* **2016**, *43*, 370–379. [CrossRef]
30. FIFA Futsal Coaching Manual. Available online: https://cdn1.sportngin.com/attachments/document/000a-2348966/FIFA_futsal-coaching-manual.pdf (accessed on 6 October 2023).
31. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1995**, *2*, 83–97. [CrossRef]
32. Barrett, P.; Hunter, J.; Miller, J.T.; Hsu, J.-C.; Greenfield, P. Matplotlib—A Portable Python Plotting Package. 2005.
33. Ayala, A.; Cruz, F.; Campos, D.; Rubio, R.; Fernandes, B.; Dazeley, R. A Comparison of Humanoid Robot Simulators: A Quantitative Approach. Available online: https://arxiv.org/pdf/2008.04627.pdf (accessed on 20 October 2023).
34. Korber, M.; Lange, J.; Rediske, S.; Steinmann, S.; Gluck, R. Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning. Available online: https://arxiv.org/pdf/2103.04616.pdf (accessed on 20 October 2023).
35. Webots Reference Manual R2023b: Supervisor. Available online: https://cyberbotics.com/doc/reference/supervisor (accessed on 20 October 2023).
36. Crystal, S. Exclusive Comparison between WebSockets and gRPC. Available online: https://www.techunits.com/topics/architecture-design/exclusive-comparison-between-websockets-and-grpc/ (accessed on 20 October 2023).