Universidade do Minho
Escola de Engenharia

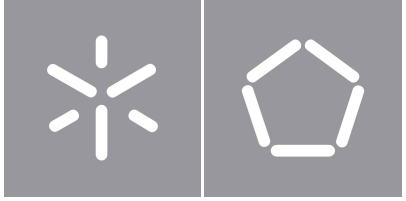
António Fernando Alcântara Ribeiro

Probability-Based Strategy for Football Cooperative Multi-Agent Autonomous Robots

Dissertação de Mestrado
Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de
Professor Doutor Luís Miguel Valente Gonçalves
Professor Doutor Agostinho Gil Teixeira Lopes

janeiro de 2024



Universidade do Minho
Escola de Engenharia

António Fernando Alcântara Ribeiro

Probability-Based Strategy for Football Cooperative Multi-Agent Autonomous Robots

Dissertação de Mestrado
Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de
Professor Doutor Luís Miguel Valente Gonçalves
Professor Doutor Agostinho Gil Teixeira Lopes

janeiro de 2024

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

The amount of knowledge acquired in this journey was only possible through hard work, motivation, perseverance and the support and incentive of people to whom I would like to thank especially.

Firstly, a special thanks to my parents, Marisa Alcântara and Fernando Ribeiro, from whom I will be eternally grateful, for the support, belief, education and opportunities they gave me. I would not have reached this point without them, so for that, thank you. To my Grandparents, António, São, Dina and Fernando, for the help and for being an example that any dream is achievable through effort and hard work. To my little brother Francisco, thank you for being a constant supply of happiness and making me proud daily. To my dear sister Inês, for the support and inspiration. To Tiago, a particular thanks to whom I had the pleasure of working alongside throughout this dissertation, for being a mentor and a role model and for pushing me to improve every day. Also, thank you to Inês Garcia for her bits of advice and constant encouragement. To my godmother Paula Gama and her family, who have always been supportive.

To my supervisors and mentors, Professor Gil Lopes and Professor Luís Gonçalves, a huge thanks for the chance to work with him and for all the help, support and guidance. To all Laboratory of Automation and Robotics members for all the fun moments not only in the laboratory but in competitions as well. Including people from the CHARMIE project, Tiago, Inês, Renata, Bruno and many more from other LAR projects such as Rui, Marco and others. To the people who showed perseverance and stayed beside me throughout this LAR@MSL project, thank you for all the nights and the fun moments and for believing in me, Joel, José, Rui, Rúben and Carolina. You are truly the best teammates one could have wished for.

A huge thanks to the RoboCup community, especially in MSL, to the Tech United team. They were one of the main reasons that this project started. Thank you for helping our team, coming to Portugal to play with us, helping us during competitions, and for all the encouragement and rooting for us. Without your help and support, this team would not have been able to play, learn and evolve as it did.

A big thanks to all Bot'n Roll members, José C., Nino, João, Pedro, Ivo, Bruno, Gonçalo, Luís, José F. and Delfim for being a great example and for supplying me with great opportunities. In particular, thanks to José Cruz for being a constant mentor for many years and for teaching me about electronics and also about life. Thank you to Nino Pereira for being a role model and helping me with the published paper.

To my close classmates, whom I thank for all the experiences we had together. To my very good friend Francisco Marques thank you for all the support. Thank you to Margarida Correia, Isaura Lopes, Rita Cunha, Renata Martins, Gabriella Sá, Ana Lages, Rubén Silva, João Pinto, Joel Costa and Bruno Silva. To my extraordinary friend Tiago Oliveira for the encouragement and support. To Abelâmio Silva, Carolina Freitas, Eduarda Cardoso, Gaspar Lopes, José Nuno, Maria Cecília, Marta Ribeiro, and Pedro Berbereia, thank you for the special moments for always being there for me no matter the situation. Also, my scoutmasters, as they were, are and always will be an example to follow and become a better person.

Words cannot express my gratitude to Carolina Lopes, who has been extremely helpful, supportive, and a true source of inspiration. Thank you for always being there, for your endless patience, for being a tremendous help, and for constantly pushing me forward for me to improve every day. Your faith in me gave me the courage to keep going no matter what. From the bottom of my heart, I thank you.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, february 2024

António Fernando Alcântara Ribeiro

Abstract

RoboCup is the Robotics World Championship and is one of the reasons for the constant growth, research and development in robotics. One of its leagues is the Middle Size League (MSL), which is a scientific football challenge with teams of five robots. The strategy of a multi-autonomous cooperative robotic system in a football game can be solved in multiple ways, commonly based on the Skills, Tactics and Plays (STP) architecture. With a different approach regarding its architecture, a centralised Probability-Based Strategy (PBS) method is presented in this dissertation and implemented in the LAR@MSL team from the University of Minho. It uses autonomous skills, generates tactics and plays in real-time, based on the probability values of every action possible. The action probabilities also position each robot to optimise the overall probability. This new method centralises all the decision-making processes in the base station, to which the robots communicate, transmit all the data acquired, and receive the ideal skill and respective arguments. The robots are still fully autonomous regarding the skills given to them, just like the strategy improvements and calibration are independent of the robots' hardware and control systems. Also crucial for this solution was the development of the simulator environment, the communication system and the base station interface. The simulator greatly improved the development time for the strategy, allowing the strategy to be developed in parallel with the robot's hardware. The communications system implemented was tested and proved reliable even in the most saturated environments. The base station interface developed is the primary debugging tool that allowed the team to interpret values in real-time and interact and calibrate both robots and the strategy. The strategy presented has been fully implemented within the team and tested in multiple scenarios, such as simulators, a controlled environment, against humans in a simulator, and at the RoboCup competition. The Probability-Based Strategy (PBS) proved modular, flexible, and easily adaptable to different game situations with minimal effort. It allows different behaviours against different opponents and can support any team size, either its own or the opponent's team, without the need for a predefined set of plays.

Keywords Football Strategy, Multi-Agent, Autonomous Robot, RoboCup, Probability-Based, Play Generator, Decision Trees, Heat Maps, Simulation, Real-Time Communication

Resumo

O RoboCup é o Campeonato Mundial de Robótica, sendo este uma das razões de constante crescimento, pesquisa e desenvolvimento da area da robótica. Uma das ligas é o Middle Size League (MSL), que consiste num desafio cientifico de futebol entre equipas de cinco robôs. A estratégia para um sistema de robôs autónomos multi-agente pode ser resolvida de várias maneiras, sendo uma das arquitetura mais comuns a Skills, Tactics and Plays (STP). Com uma abordagem diferente em relação à arquitetura, uma estratégia centralizada baseada em probabilidades foi apresentada ao longo desta dissertação e implementada na equipa LAR@MSL da Universidade do Minho. A estratégia usa habilidades autónomas e gera jogadas e táticas em tempo-real baseando-se nos valores de probabilidade de todas as ações possíveis. As probabilidades de cada ação também são utilizadas para o posicionamento dos robôs, de forma a otimizar a probabilidade total. Este novo método centraliza todas as decisões no computador central, e é para este que os robôs comunicam, transmitem os dados adquiridos e recebem a habilidade ideal e respetivos argumentos. Os robôs continuam a ser completamente autónomos no que toca às habilidades e qualquer melhoria à estratégia e respetiva calibração são independentes do hardware e sistemas de controlo dos robôs. Igualmente importante e também apresentado nesta dissertação são o ambiente de simulação, o sistema de comunicações e a interface do computador principal. O simulador melhorou drasticamente o tempo de desenvolvimento da estratégia, permitindo que esta conseguisse ser desenvolvida em paralelo com o hardware dos próprios robôs. O sistema de comunicações foi testado e provou-se fiável mesmo em ambientes saturados. A interface do computador principal foi desenvolvida como uma ferramenta de *debug* e permitiu a equipa interpretar valores em tempo real, bem como interagir e calibrar os próprios robôs. A estratégia aqui apresentada foi também totalmente implementada na equipa e testada em vários cenários, como simuladores, num ambiente controlado, contra humanos num simulados e ainda na competição RoboCup. A Probability-Based Strategy (PBS) provou ser modular, flexível, e facilmente adaptável a diferentes situações de jogo com pouco trabalho. Permite à estratégia ter comportamentos diferentes contra diferentes oponentes e é independente do número de robôs, quer da própria equipa quer da equipa oponente, sem haver a necessidade de jogadas predefinidas.

Palavras-chave Estratégia de Futebol, Multi-Agente, Robô Autónomo, RoboCup, Baseado em Probabilidade, Gerador de Jogadas, Árvores de Decisão, Mapas de Probabilidades, Simulação, Comunicações em Tempo Real

Contents

List of Figures

List of Tables

List of Equations

Acronyms

API Application Programming Interface

BMS Battery Management Systems

CPU Central Processing Unit

FIFA Federation Internationale de Football Association

GPU Graphics Processing Unit

gRPC Google Remote Procedure Calls

GTK+ GIMP ToolKit

GUI Graphical User Interface

I2C Inter-Integrated Circuit

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

JSON JavaScript Object Notation

LAR Laboratory of Automation and Robotics

LiDAR Light Detection and Ranging

MSL Middle Size League

ODE Open Dynamics Engine

OSI Open Systems Interconnection

PBS Probability-Based Strategy

PCB Printed Circuit Board

PIC Peripheral Interface Controller

PID Proportional – Integral – Derivative

PLA Polylactic Acid

RAM Random-Access Memory

RGB Red-Green-Blue

RGBD Red-Green-Blue-Depth

ROS Robot Operating System

RTDB Real-Time DataBase

SDL Simple DirectMedia Layer

STP Skills, Tactics and Plays

TCP Transmission Control Protocol

TOF Time Of Flight

UDP User Datagram Protocol

Chapter 1

Introduction

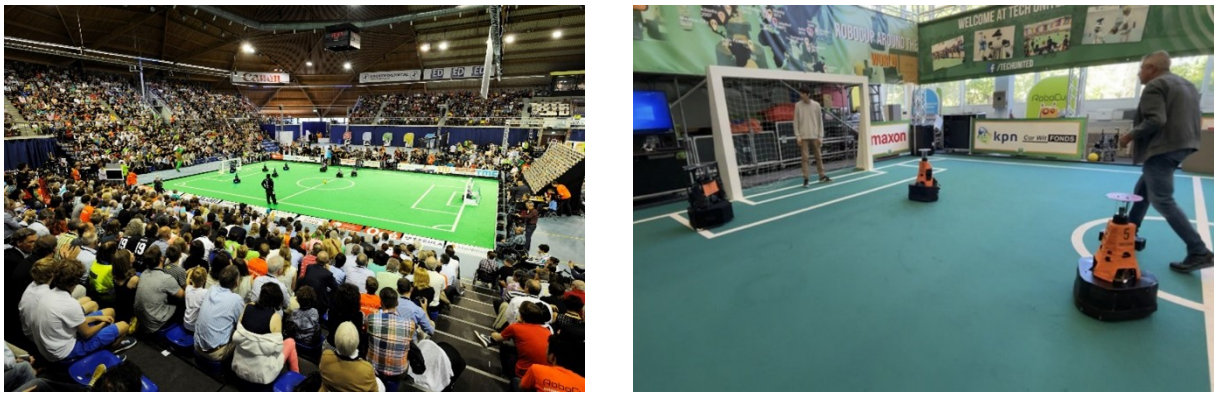
Robotics is one of the fastest growing areas, and it is the field of computer science and engineering that handles the design, assembly and management of robots. Robots have multiple designs that vary based on the task they are made to do. Robots can be classified in multiple ways based on their shape, size, function, environment, degree of automation and mobility.

This dissertation focuses on fully autonomous mobile robots with the objective of playing football. These robots are not industrial, for service, medical use, or even military applications, but are the key to getting people interested in robotics and investing in its research and development. Various scientific contributions and even technology standards, like the EtherCat, were developed because of the need for that technology in football robots.

1.1 Problem Description

RoboCup started in 1997, and leagues have become more competitive over the years based on the motivation of all teams to research and develop robots. The Middle Size League (MSL) is no exception [1]. MSL is one of the main leagues and has a large amount of areas present. Mechanics, Electronics, Programming, Control Systems, Multi-Agent cooperation, Communication, Artificial Intelligence, Computer Vision, Strategy, Decision-Making, Human Interaction and much more are the foundation of MSL.

One of the RoboCup goals is to play against the world champion human football team with the best MSL team in 2050 [2]. Teams with five fully autonomous robots play against each other in every MSL game, and teams are free to design their hardware and software while complying with the league rules [3]. Figure 1 shows two MSL showcases, the final game from RoboCup 2013 in Eindhoven (a) and a demonstration from the team Tech United (b).



(a) (b)

Figure 1: MSL showcases: (a) Final in RoboCup 2013 [1]; (b) Tech United's demonstration

The University of Minho and the Laboratory of Automation and Robotics (LAR) first participated in MSL in 1999 with the name "MinhoTeam". MinhoTeam participated in multiple leagues and already won multiple awards. There were already various MinhoTeam generations [4, 5, 6] in the last 25 years, and one of the objectives of this dissertation, together with other team members, is to restart MinhoTeam, now with the name LAR@MSL [7].

Many existing solutions for coordinating multi-robot football teams have some limitations. A common decentralised approach can lead to synchronisation issues between robots, which are not ideal in real-time problems. Solutions relying heavily on predefined playbooks restrict the flexibility and adaptability of the robot team. Playbooks require a lot of manual setup time and make it difficult to easily parameterise behaviours or add modularity. These solutions depend on having carefully crafted plays that cover every scenario the robots may encounter. However, it is difficult to account for all possible situations with predefined plays.

Overall, the dependency on decentralisation, playbooks, and predefined plays and tactics limits the coordination ability of teams. These approaches lack the flexibility, synchrony, and adaptability required for robust multi-robot coordination in a complex and dynamic environment such as a robot football game. There is a need for a solution that provides centralised supervision, while still allowing decentralised execution, maintaining the robot's autonomy. Modular and parameterised behaviours would increase adaptability. An approach combining these strengths and solving the common strategy problems would very much benefit a team.

1.2 Motivation

The robotics trend is to work as teams of robots or multi-agent cooperative systems to complete more complex tasks, with full control in real-time and efficient interfaces. To achieve cooperation and collaboration amongst robots a good communication system is vital. A good task-planning platform is essential to prepare robots for those complex tasks, making simulators one of the best methods. To organise and manage/control such complex systems, it is necessary to have an integrated and centralised software platform. This platform must allow a very simple yet comprehensive interface, be a good debugging tool and have the capability to log and represent real-time data.

The RoboCup MSL is a good test bed and benchmark to prepare for that challenge, as it incorporates all the issues above described. An MSL team has five robots that have to cooperate, maintain synchronism in a real-time application, require a good and efficient communication system, and depend on a central management control software with a comprehensive interface capable of representing real-time data. All the described challenges are the main reasons for this dissertation's motivation.

Motivation is also based on the will to be part of the research community [8], especially to contribute scientifically to this major league with new and creative solutions. This research community has already developed new products, created large robotics companies, and sponsored big events, among others.

On top of that, the teams that participate every year in MSL and RoboCup are a big inspiration and have a lot of achievements and creations in the area of Robotics. Interacting with those teams allows the team to communicate and exchange ideas with people from world-renowned universities like the Eindhoven University of Technology and Carnegie Mellon, as well as some companies like Philips, VDL and even ASML, adding personal overall knowledge in robotics.

1.3 Objectives

This dissertation goals reside on the following significant topics:

- Creation of a centralised game strategy architecture capable of maintaining real-time synchronism between all robots;
- Make the game strategy parameterisable, modular and flexible for easy adaptability between game and opponent teams;
- Eliminate the need for a playbook, a predefined list of plays and tactics, in order to keep the play solutions dynamic and reduce the development time;
- Development of a reliable and efficient communication system capable of performing under highly saturated environments;
- Development of a simulation environment in order to accelerate the strategy development time and capable of maintaining an easy transition between the simulation and the real-world scenarios, in order to have a hardware-independent system;
- Creation of a debugging tool capable of receiving referee instructions and representing game data in real-time;
- Actively participate in robotic football competitions in order to test and acquire data for further analysis, such as "Festival Nacional de Robótica" and RoboCup;
- Results demonstration, description and analysis, and comparison to the current game strategy solutions.

Strategy, Communication and Simulator are the focus of this dissertation. Because the team has five robots and the information gathered is restricted by each robot's sensors, communication is critical for the robots to cooperate. Besides the three main parts of this dissertation, strategy, communications and simulation, the robot's hardware and base station are presented as context and are necessary to the team and to the dissertation objectives.

The creation of a game strategy method, centralised and synchronous, based entirely on probabilities that generate the plays in real-time, eliminating the need for a playbook can solve the current game strategy problems. Adjusting the probability formulae used can change completely a team's intention and behaviour

on the field resulting in an easy parameterization. In order to achieve the objectives regarding the strategy, the solution would be based on computing the probability of every possible action and choosing the options with the best outcome. Based on the probabilities, all the decision-making processes would occur in the central computer and in real-time based on the communication system [4]. Since the strategy is based on the probability of outcomes, its modularity would be based on the probability of actions and could be adjusted based on the opponent team and other field elements.

The communications system would need to be reliable, fast and efficient, and needs to be based on a wireless network specified in the league rules [3]. All communications must occur in the same network; no two robots can communicate outside the game's access point. With a few exceptions, almost all teams use a User Datagram Protocol (UDP) approach to communicate with every robot [9].

Since hardware dependency could be a problem during this dissertation, a simulation tool is also essential and part of this work. Considering the amount of code and strategy to be developed and tested before having functional robots, a simulator helps the team fulfil the timeline available for the competition. Also, the simulation tool will be essential to recreate games and develop code allowing the use of log files acquired from previously played games [5].

As important as the simulator, the base station, required by the rules, has a very significant role in the whole system. Its objective is to represent what the robots are doing to facilitate debugging the robot's behaviour and to check its state and information. During games, the base station is the primary tool to have information in real-time about the robots and the game.

1.4 Document Structure

The first chapter, Introduction presents the problem description, motivation, and scientific contributions. Chapter two includes a literature review summarising the state of the art in areas like game strategies from multiple teams, team simulators, communication systems and base stations. Chapter three, Theoretical Foundations, presents a brief introduction to the simulator software used, as well as explanations for Sockets, Log Probabilities, Normal Distribution Curves, Heat Maps, Decision Trees, Graphs, Pathfinding algorithms and the Hungarian algorithm.

The fourth chapter dives into the robot's hardware presenting it as context for the real-world results. The communication system is presented in the fifth chapter. In chapter Simulator, the sixth chapter presents the simulation environments developed as well as its its functionalities. The seventh chapter is the core of the dissertation and presents the game strategy decisions that are made based on probabilities of every

possible action, the play generator, the heat maps and all the parts within the game strategy. The team's base station interface is presented in chapter eight.

The ninth chapter presents the tests and results. It analyses the communication system performance, strategy effectiveness, simulator validation, and real-world testing. The key findings are summarised, and the advantages and limitations are discussed. In the last chapter, the conclusions, summary and objectives of this dissertation are discussed.

Finally, the appendix provides supplementary information about some hardware, mechanical and electronics developed to improve the robot's overall performance.

Chapter 2

State of the Art

In MSL, each team plays football with five autonomous robots that cooperate for the common goal of scoring goals. The league rules are based on Federation Internationale de Football Association (FIFA) with amendments for robots, which are released yearly to advance the state of the art of intelligent robots [3]. The teams are free to develop and use any hardware and software they want, having to cope with a few restrictions defined by the official league rules. This dissertation intends to solve problems in different areas, and therefore, this state-of-the-art needs to comprehend multiple areas and subjects and is organised on the following topics.

- Strategies;
  - Skills, Tactics and Plays (STP);
  - Tech United's Approach;
  - Machine Learning Methods;
  - Play Transition Methods for STP;
  - Alternatives to STP;
- Team Simulators, Communications and Base Station;
  - Tech United Eindhoven;
  - Water;
  - CAMBADA.

2.1 Strategies

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

2.1.1 Skills, Tactics and Plays (STP)

The Skills, Tactics and Plays (STP) architecture was initially developed by the team CMDragons [10, 11, 12], 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 [13], to overcome their main issues with the algorithms used by then. As the name suggests, the STP is divided into three parts: Skills, Tactics and Plays. This section explains how each component works and generates the desired output when working together.

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 [13].

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 for each tactic. Tactics within this framework are also parameterised. Examples include defending the goal, executing a pass, intercepting the opponent's pass, and assisting in an attack [10, 11, 12].

Plays

Building upon the tactics, the plays delineate each robot's specific roles and sequential actions. 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 behaviour 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 exit conditions are likewise fulfilled.

Playbook

As a rule of thumb, teams have multiple playbooks to 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 generalised, depending on a team's needs and preferences. If one assumes that a single play covers all possible game states, 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 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 [13].

2.1.2 Tech United's Approach

Tech United's team was chosen for this study because of its full open-source and well-documented code. Their game strategy uses the STP architecture in addition to a few different modules to complement it. Each robot creates a world model in real-time with the information gathered from the other robots. This world model is the football pitch with the position of all robots on the field and the ball. This world model is also represented in the base station previously described [14].

After the world model is generated with all the information gathered, the decisions are taken by "The Defense/Attack Control System" (*Defcon*), which is a portion of code responsible for translating all the information to a game state. Based on the *Defcon*, the robots decide either to attack or defend. After that decision, the Role Assigner handles the actual strategy selection. The Role Assigner, as the name suggests, assigns roles to every robot. These roles are decided democratically by the other robots.

After each robot's role has been determined, it selects the exact task it should do from a previously defined set [15]. These tasks consist of simple robot actions or skills, for example, defending the ball from an opponent, passing the ball to another robot or kicking the ball to the goal. In summation, the strategy

is organised in a STP architecture. This approach facilitates long-term tactics, preventing *Defcon* from breaking up the strategy after every action. Figure 2 shows a representation of the STP model [13].

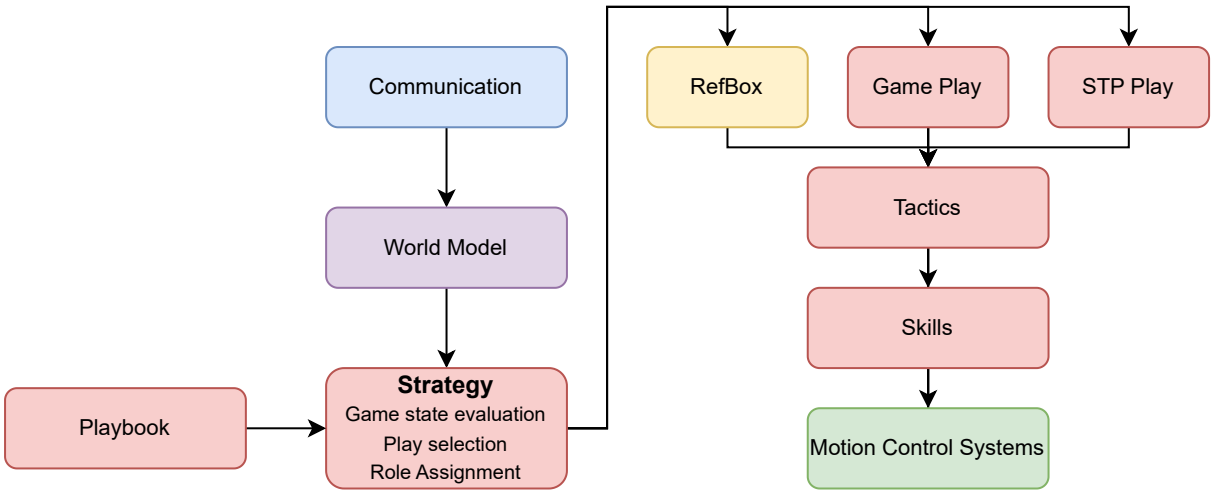


Figure 2: Tech United STP schematic and strategy integration [13]

2.1.3 Machine Learning Methods

Diverse approaches were employed in developing the skills by other teams, such as utilising reinforcement learning [16, 17]. However, the decision-making aspect continued to rely on traditional methods within the STP architecture [18]. Because of the STP dependence, a previously created playbook that addresses various potential game situations should exist.

Some machine learning algorithms were also used to transition between different plays or strategies. For example, a Graph Evolution, developed in 2014 [19], 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.

2.1.4 Play Transition Methods for STP

The choice of which play to choose given a particular game state was solved with Bayesian Classifiers in 2018 [20] 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 [21], but its implementation was based on parameterising and deciding the desired play and not generating them in real-time, as is the case in this dissertation.

2.1.5 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 [22, 23], applied within a simulation environment. The downside of having a full reinforcement learning implementation is the lack of modularity and parameterization, which are important when playing against different teams in a short period. When a new opposing team employs different behaviours, Reinforcement Learning algorithms can lack flexibility and speed when adopted. Implementing changes often requires more time and computational resources compared to hand-coded strategies. Classic strategies can allow behaviour changes on the fly during a match (during the break) or in between matches. Some simulation league teams use genetic algorithms [24], which do not rely on the STP architecture. However, these have other limitations, such as modularity constraints and training requirements.

2.2 Team Simulators, Communications and Base Station

There are many teams in MSL, but this section focuses only on three of them. The choice of these three teams is based on a few key points. The first one, Tech United Eindhoven, is the world champion and one of many teams that has everything open-source. They think and invest in new teams for the league. Water is another strong team with innovative ideas. Their approach to the problems very rarely uses a common solution, so their different ideas were one of the reasons this team was considered in this study. The third and final team is CAMBADA, a Portuguese team from Aveiro, and they were already World Champions in 2008. Other Portuguese teams participated over the years, for example, 5DPO [25], ISePorto and ISocRob.

2.2.1 Tech United Eindhoven

Tech United Eindhoven is a team from the Netherlands representing the Eindhoven University of Technology from the city of Eindhoven. This team started in 2005 and first participated in a competition in 2006. They achieved the world champion title several times in 2012, 2014, 2016, 2018, 2019, 2022 and 2023 [26]. They call their Robots "turtles", but each has a specific name from famous football players or tech jokes. It is important to highlight their fair play as their achievements, research, software and hardware are open-sourced [27]. Figure 3 shows the Tech United robots or, as they call them, "turtles" [28].

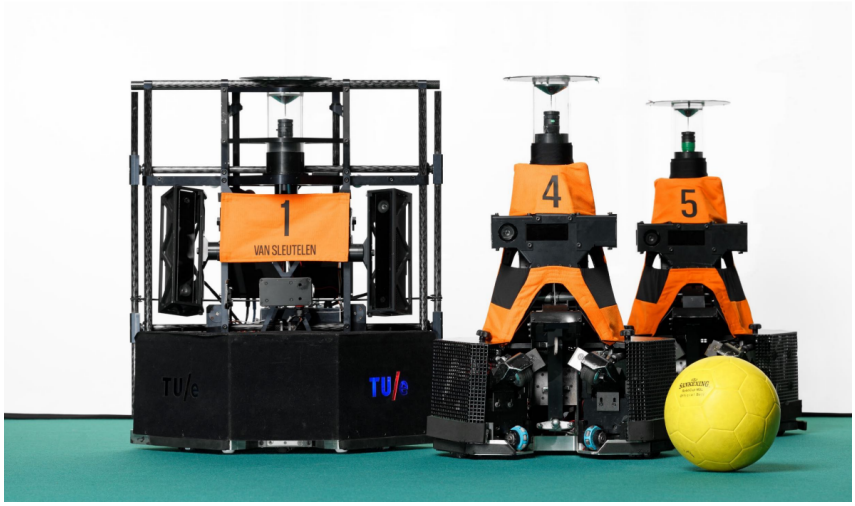


Figure 3: Tech United team robots [14]

Simulator

Nowadays, the Tech United team does not use a simulator very often. The free access to a field and fully functional robots allow them to test without the need to simulate. However, they still use it to replay virtually previous games from official competitions, as all the data is logged [15]. Their simulator was developed using MATLAB. Figure 4 shows the Tech United simulator showing the computation of the desired position for each robot.

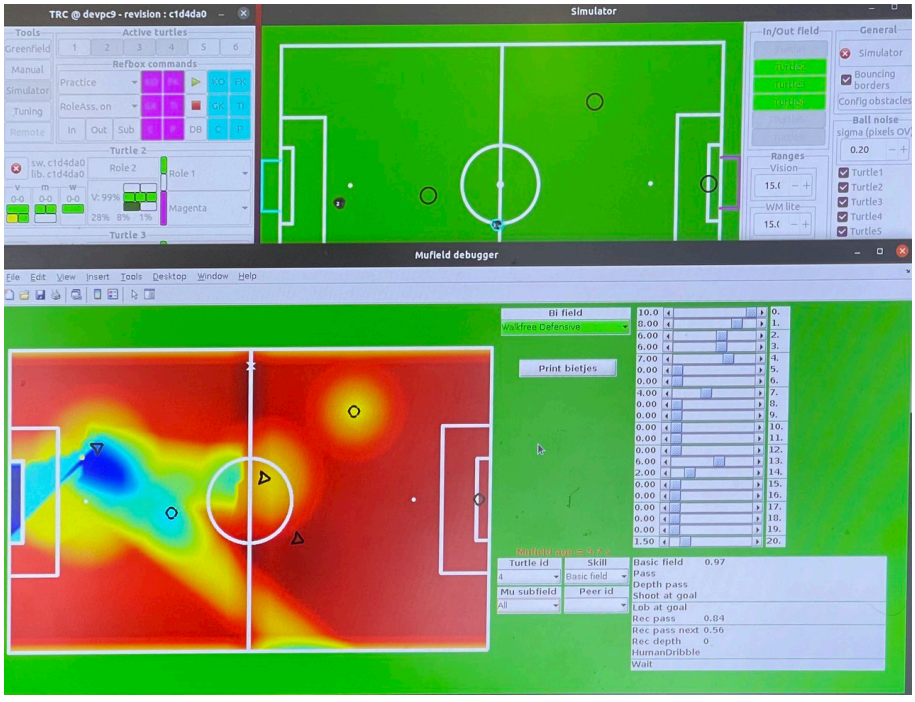


Figure 4: Tech United simulator environment

Communications

For communication, the team uses the same method as most other teams. It is called Real-Time DataBase (RTDB) and was first developed in 2004 by a Portuguese team called CAMBADA [29]. Its development solved the synchronization and data-sharing issues common on all teams of the league. Since then, because of the open-source spirit, other teams have used and helped improve it over the years. It was originally written in C++, but now, a Python 3 version is also available. Their communication system uses an UDP multicast communication between the robots and the base station Computer.

Base Station

The base station is the main computer that links the robots from each team to the Referee Box. Tech United's base station is called "Turtle Remote Control" and was also developed in MATLAB. They had a previous version developed in C with Glade and the GIMP ToolKit (GTK+) frameworks, but they rebuilt it in MATLAB. This software is used during games, demos and even to test different strategies and configurations. It handles the communication, the live representation of the game state and even the Referee Box (RefBox) commands. It represents all the information from the "turtles", in real-time, such as battery charge, Central Processing Unit (CPU) usage, role in the game, vision system and even communication loss rate. Tech United base station screen is shown in Figure 5.

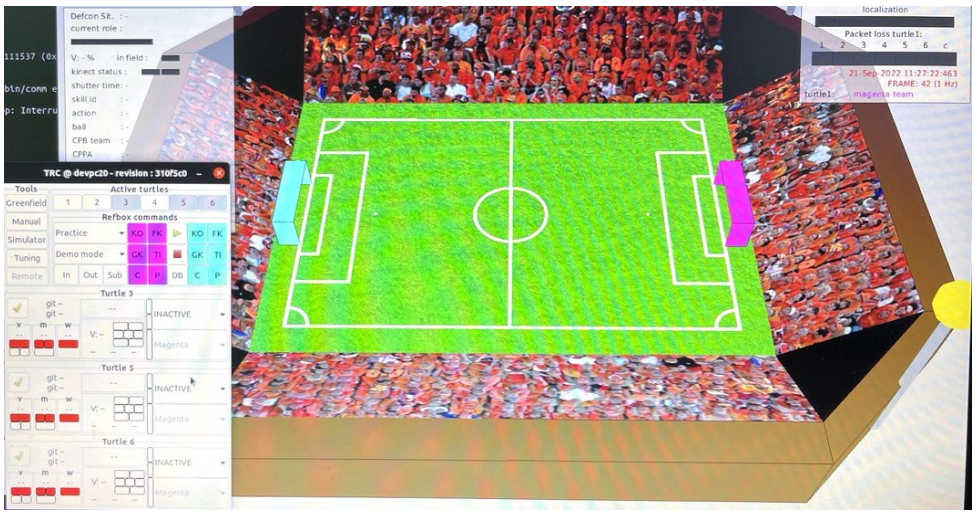


Figure 5: Tech United base station with 3D representation

2.2.2 Water

Water is a Chinese team from the University of Technology in Beijing and has participated in national competitions since 2006 [30] and in RoboCup since 2010. They were World Champions five times in 2010, 2011, 2013, 2015 and 2017. The robots of the team Water are shown in Figure 6.

Figure 6: Water team robots

Simulator

In contrast to most of the other MSL teams, team Water does not use any simulator. They use real robots to develop everything. The advantage is that there are no problems transitioning from the simulation to the real world. The disadvantage is that they cannot work offline and always need a field and functioning robots. MSL robots are manually developed, requiring considerable maintenance, making it difficult to have the full team working all the time.

Communications

Just like the simulator and the choice of the operating system, their communication system is also innovative. They are the only team using the Transmission Control Protocol (TCP) protocol [30]. They establish a client/server approach, being the Water Coach (base station) as the server and each robot as a client. Figure 7 represents the communications scheme the team Water uses.

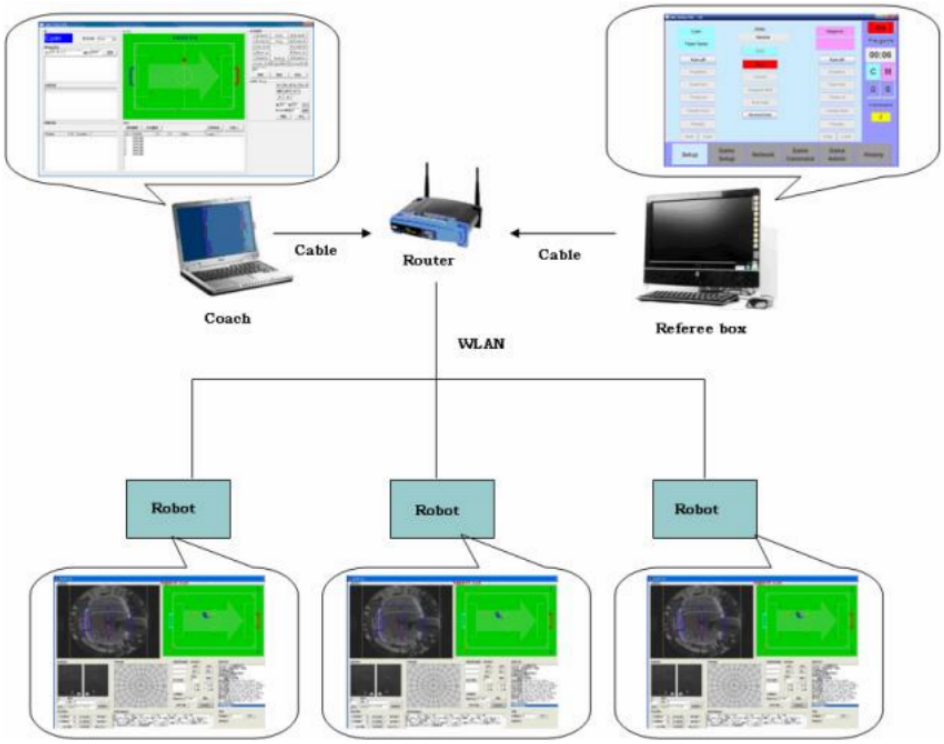


Figure 7: Team Water communication schematic [30]

Base Station

Their base station is called "Water Coach" (Figure 8) and it was developed using Microsoft Visual Studio and written in C++. Water is one of the very few teams that use Windows on their robots. Most other teams use Linux with Ubuntu distributions. Just like Tech United, their base station serves the purpose of communicating with the robots and handling the Refbox events.

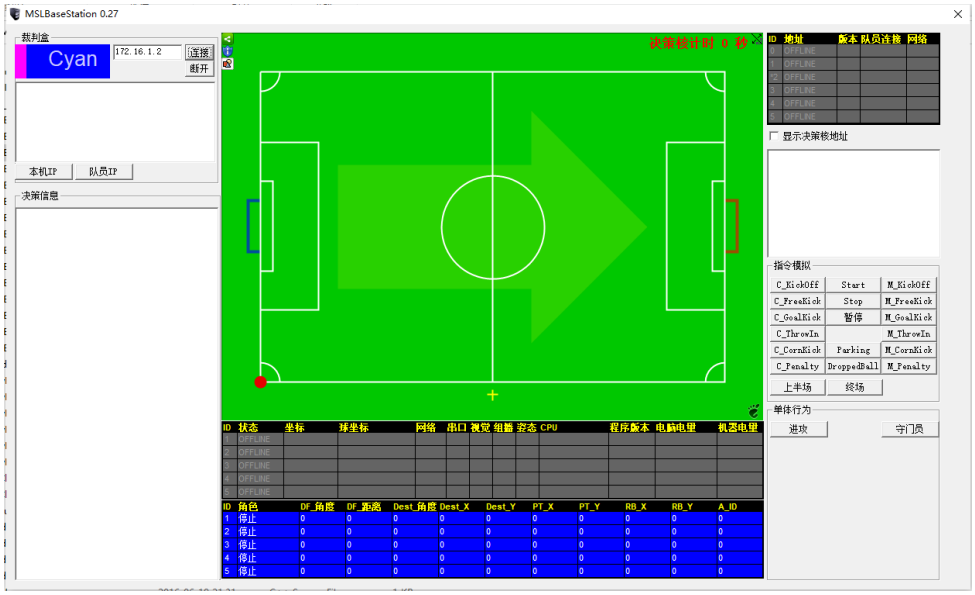


Figure 8: "Water Coach" - Water base station

2.2.3 CAMBADA

CAMBADA is an acronym that stands for Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture and it is a MSL Team from the University of Aveiro in Portugal. The team was created in 2003 and won the RoboCup in 2008. They are the Portuguese Champions since 2007 [31]. Unfortunately, due to the pandemic situation, the team has ended its activity. However, they were very active and helped to develop this league in many areas. CAMBADA team and their robots are shown in Figure 9. Just like Team Water, CAMBADA does not use a simulator, as it has free access to a full-size MSL field.

Figure 9: CAMBADA team

Communications

For communications, team CAMBADA started to develop the solution most used in MSL called RTDB in 2004 [29]. They communicate using the same protocol as Tech United, the UDP multicast, and then use RTDB to manage, clean, fuse and synchronise all the data in real-time. With RTDB they have the same real-world model between all robots without surpassing the bandwidth limits imposed by the RoboCup rules [3].

RTDB's major advantage is the synchronised data between robots with the shortest communication time for real-time applications [29]. Although other solutions have been used in RoboCup, this method enforces a correct timeline, meaning that refreshing information items and the temporal accuracy of the data stored in the RTDB has higher feasibility. Figure 10 shows the RTDB architecture and its shared memory.

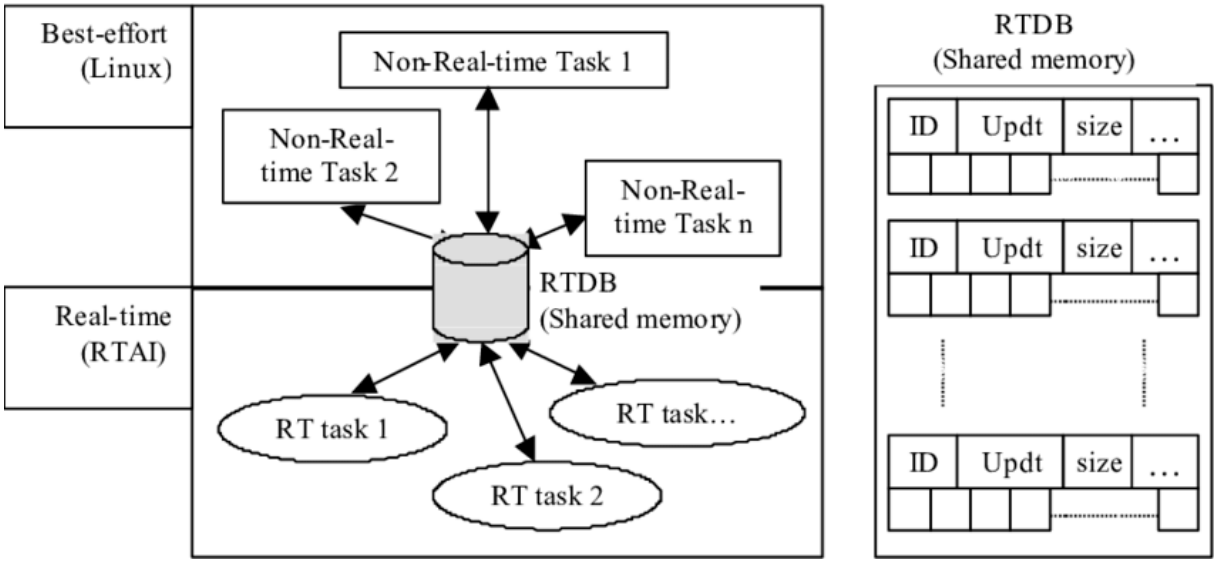


Figure 10: Architecture of the RTDB being accessed by local processes [32]

Base Station

CAMBADA base station, Figure 11, is used to control and monitor their robots. It was developed in C++ language using the Qt framework. Even though Qt runs on multiple operating systems, this team uses it under Linux. They can set roles and game states and monitor most of the robot information, such as the battery charge, behaviour, and even RTDB debug flags.

Figure 11: CAMBADA base station

Chapter 3

Theoretical Foundations

As previously described, the core of this dissertation focuses on probabilities applied to robotics football strategy. As such, a theoretical foundation is presented to discuss the necessary topics from the following chapters. The topics presented are:

- Robotics Simulators;
- Sockets;
- Log Probability;
- Normal Distribution Curve and Bayesian Curve;
- Heat Maps;
- Decision Trees;
- Graph;
- A* Path Finding Algorithm;
- Hungarian Algorithm.

3.1 Robotics Simulators

Most teams in the MSL league choose to develop their simulator, but many generic robotics simulators can be used. A robotics simulator is a software capable of creating simulation environments to study, research, and develop platforms without using hardware [5]. On the other hand, there is a disadvantage to using simulators, as the transition from the simulated environment to the real-world is not easy and it

requires some adaptation of code to be able to behave similarly. Some simulator software was considered, but the software used in this dissertation was Webots.

Webots is a 3D graphical robotics simulator developed by Cyberbotics, a Swiss enterprise that started in 1996. It was a paid license and relatively expensive for many years, requiring a physical USB key to run. In 2019, it became open-source. However, it still has paid licenses for professional use, giving access to 24/7 support and consultancy. It uses the same physics engine as Gazebo, Open Dynamics Engine (ODE) but runs on Linux, Mac and Windows. An example of a simulation environment developed in Webots with an Aibo robot is shown in Figure 12.

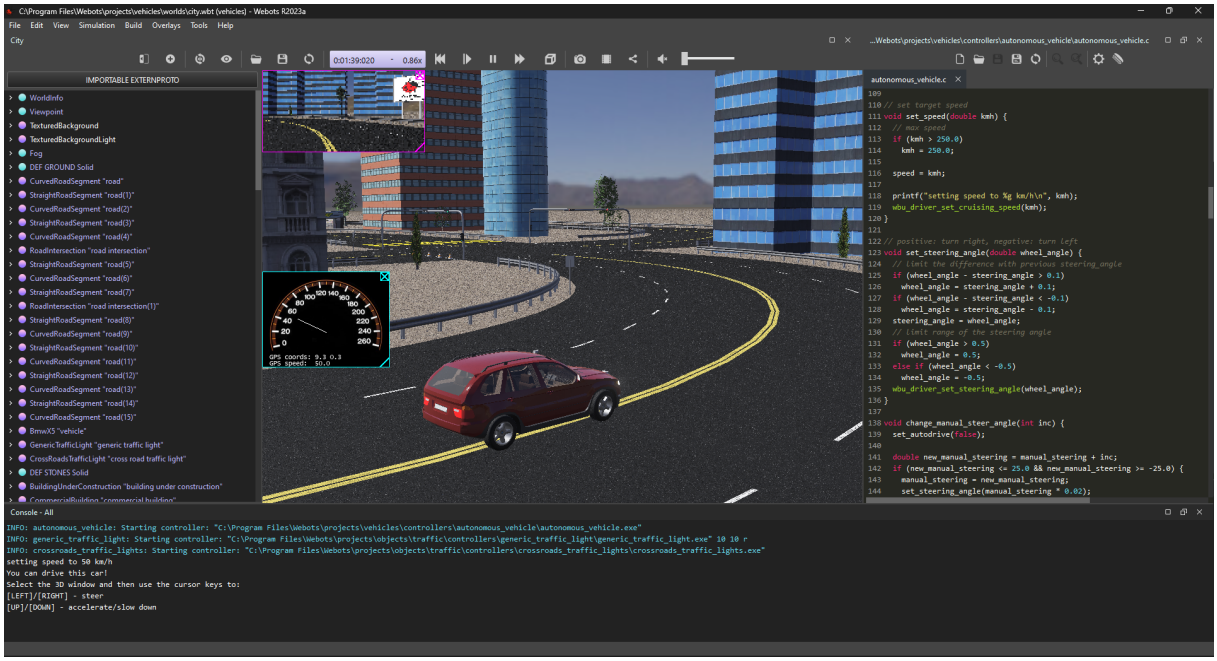


Figure 12: Example of simulation in Webots

Webots is ideal for digital twins, and besides an API, the communications between any code and the simulation can be based on TCP. Robot controllers can be written in C/C++, Python, Java and MATLAB, but they also support ROS. It allows external controllers to actuate and get information from the simulation in any language as long as the TCP/IP protocol is used to communicate between the internal and external controllers.

It also offers a Web Interface that easily shares, runs and interacts with simulations from the cloud. The simulation runs on the computer or the cloud, shown in the web interface. Users can fully control the environment from this web interface and even program the robots directly.

The main fields of applications are:

- Fast Prototyping;
- Wheeled and Legged locomotion;
- Suitable for Air, Water and Land Robots;
- Multi-Robot Simulations;
- AI and genetic algorithms;
- Computer vision.

When selecting a simulator various factors must be considered such as performance, community, documentation, and stability. Performance-wise, Webots was one of the best options due to the low usage of CPU, GPU, memory and disk when compared to the most common alternatives [33, 34]. Since it became free and open source, Webots has grown and has been very active.

Gazebo is widely used but most of the options available regarding communication were based on ROS, something that the team had no intent of using. CoppeliaSim has very poor performance compared to Gazebo and Webots, especially when the simulation environment is a multi-robot system, which is the case [33, 34].

3.2 Sockets

Sockets are a method of communication either on the same device or over a network. They link two processes through a connection defined by an Internet Protocol (IP) address and the port numbers on each side. They can communicate both ways and be either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). They work on the transport layer of the Open Systems Interconnection (OSI) model [35]. Figure 13 shows an example of a socket between two machines.

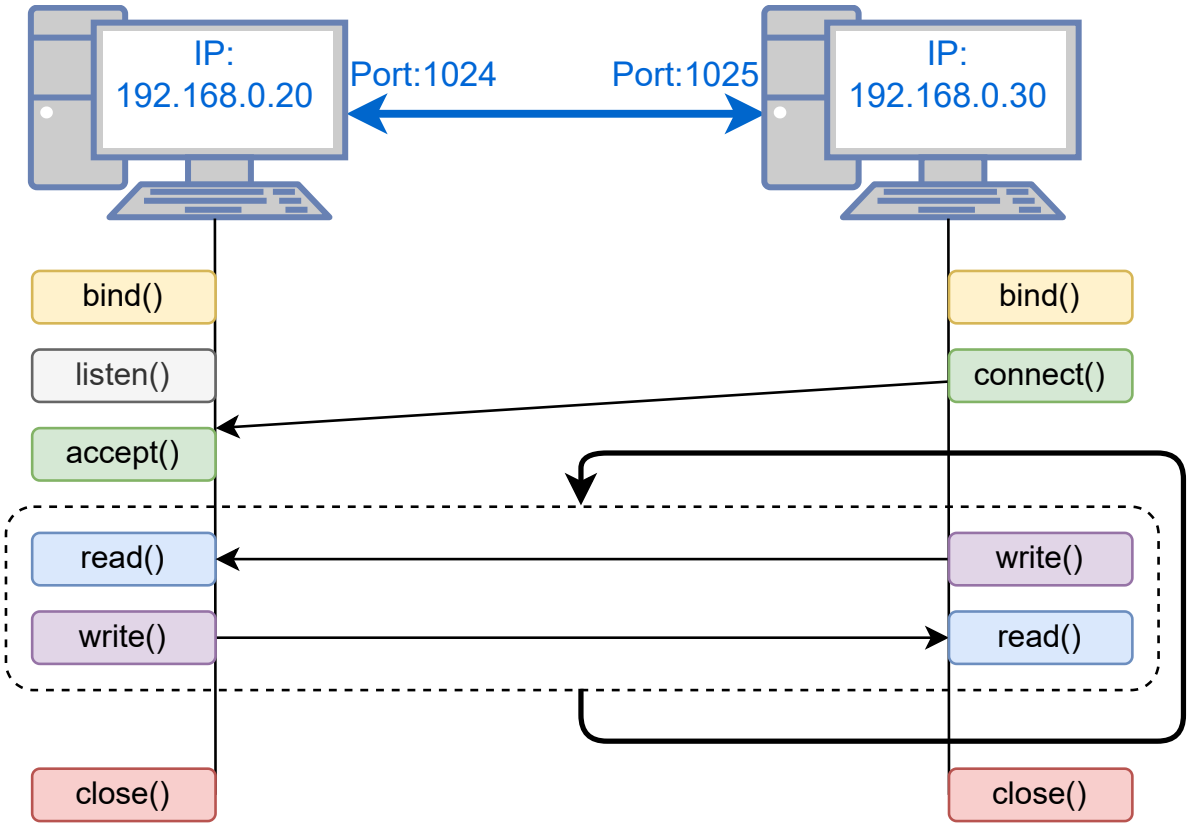


Figure 13: Socket connection between two machines

Initially, the two machines bind a socket to a port number. One of the machines connects to the other, and as soon as the connection is accepted, the two machines can communicate until the socket is closed.

3.3 Log Probability

The logarithm of a probability is called "Log Probability", and it transforms probabilities into an arbitrary value representative of that probability using Equation 3.1. This equation changed the probability values into a logarithmic space using different limits [36], visible in Equation 3.2.

$$P(x)' = log(P(x)) \tag{3.1}$$

$$\begin{aligned} 0 \leq P(x) \leq 1, \text{ Probability Space} \\ -\infty \leq log(P(x)) \leq 0, \text{ Logarithmic Space} \end{aligned} \tag{3.2}$$

This approach has two significant advantages, especially in computer science terms: speed and accuracy.

3.3.1 Speed

When using probabilities, it is common to calculate the probability of A and B. This is calculated by multiplying the two probabilities, but in the logarithmic space, the multiplication becomes an addition, shown in Equation 3.3.

$$\begin{aligned} P(A \wedge B) &= P(A) * P(B), \text{ in Probability space} \\ log(P(A \wedge B)) &= log(P(A) * P(B)) \\ &= log(P(A)) + log(P(B)) \\ &= P(A)' + P(B)', \text{ in Logarithmic space} \end{aligned} \tag{3.3}$$

Since multiplication is more demanding of computational power when compared to addition, this formula is often faster when the amount of multiplications is considerable. The conversion of spaces is demanding but is only required once.

3.3.2 Accuracy

The increased space range in the logarithmic form solves other computational problems and improves the accuracy of the probabilities when working with very small numbers. Computers need to approximate some real values when working with small numbers, which can often generate a floating point problem. The use of Log Probabilities also helps with this problem because very small probabilities become a more workable value to computers [36], visible in Equation 3.4.

$$\begin{aligned} P(x) &= 2.345 * 10^{-678} = 0.000000(...)2345, \text{ in Probability space} \\ log(P(x)) &= 2.345 * 10^{-678} = -677.629, \text{ in Logarithmic space} \end{aligned} \tag{3.4}$$

3.4 Normal Distribution Curve and Bayesian Curve

Normal distribution curve is a symmetric continuous distribution, the most used model in a Bayesian linear regression. This curve has two key parameters, σ and μ, and is visible in Equation 3.5. This curve is used in robotics when a centred value is wanted, and its variation is the same in either direction [37].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.5}$$

This equation has its data centred on the average, which is defined by the Greek letter μ and its standard deviation is defined by the letter σ. The standard deviation controls the width of the "bell" shape. The influence of these two parameters is visible in Figure 14 where the value μ moves the centre of the curve and the value σ changes the size of the "bell".

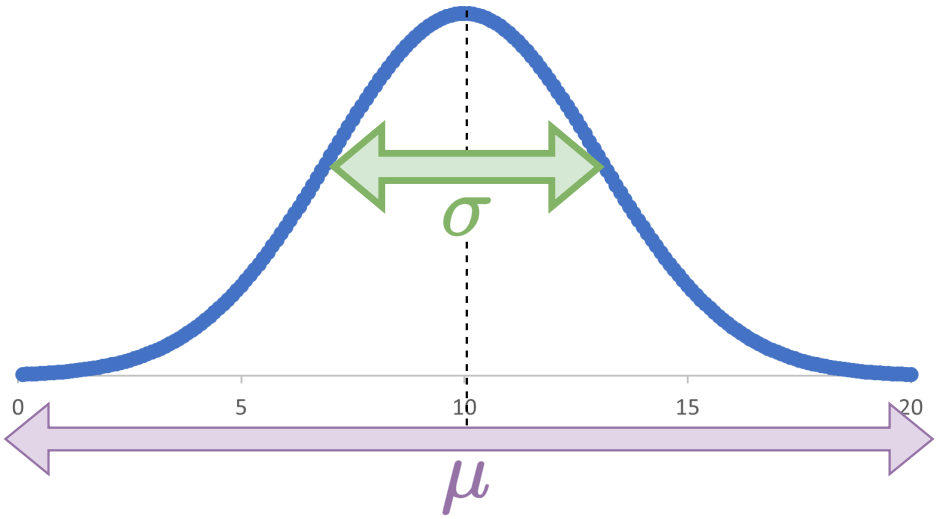


Figure 14: Influence of parameter μ and σ in a normal distribution curve

In this dissertation, the term "Bayesian curve" refers to a normal distribution curve. However, it is called a Bayesian curve because even if the probability formulae in the Game Strategy chapter are changed, the curve still remains a Bayesian curve, although it may not be a normal distribution curve anymore. This can happen if, with new probability formulae, the curves stop being symmetric or have more than one peak.

3.5 Heat Maps

Human perception of data and representation tools are extremely important in computer science, robotics and control systems as they are the key to better understanding data. Data can be presented by graphics, tables and other methods. Heat Maps use colours to represent data and are widely used in websites, geographical data and motion planning algorithms [38]. This representation makes it easy to visualise complex data in real-time and drastically reduces the time required by a human to interpret it. Figure 15 shows a heat map with the matrix values and their respective colour.

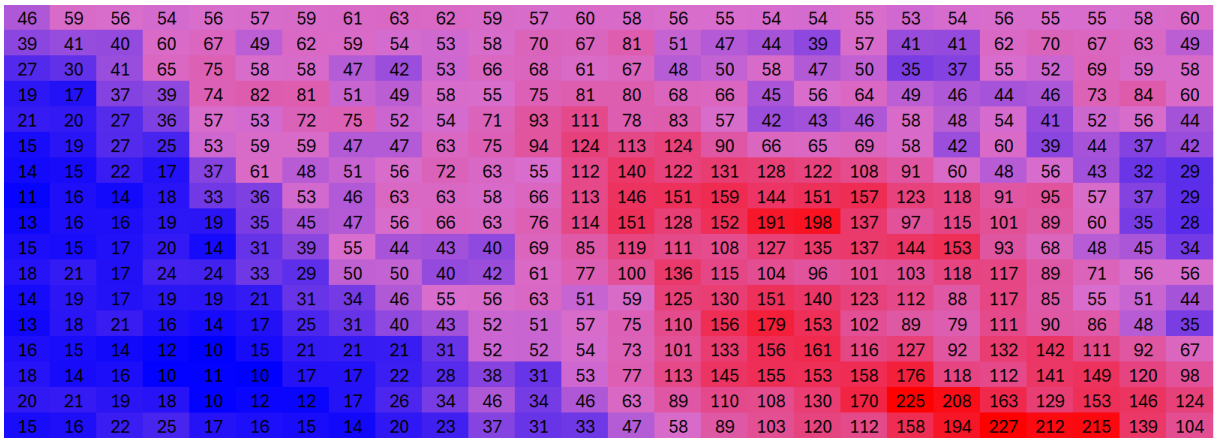


Figure 15: Example of heat map

3.6 Decision Trees

Decision trees are one decision-making method with a structural representative diagram. Starting from the initial root, each branch represents a possible condition, and at the end of each branch, there is a node that diverges into multiple other branches. At the end of a possible path, there are the outcomes of that path. So, each path is defined through a group of multiple variables, each variable associated with a node [39, 40].

Its visual representation helps break down complex decision processes, and its tree-shaped representation makes it intuitive for humans. It is also a commonly used supervised machine learning algorithm, mostly for classification and regression tasks. Figure 16 shows a simple diagram of a decision tree with the root, nodes with respective conditions/variables and end paths with each outcome. Conditions are visible in blue and possible outcomes are presented in yellow.

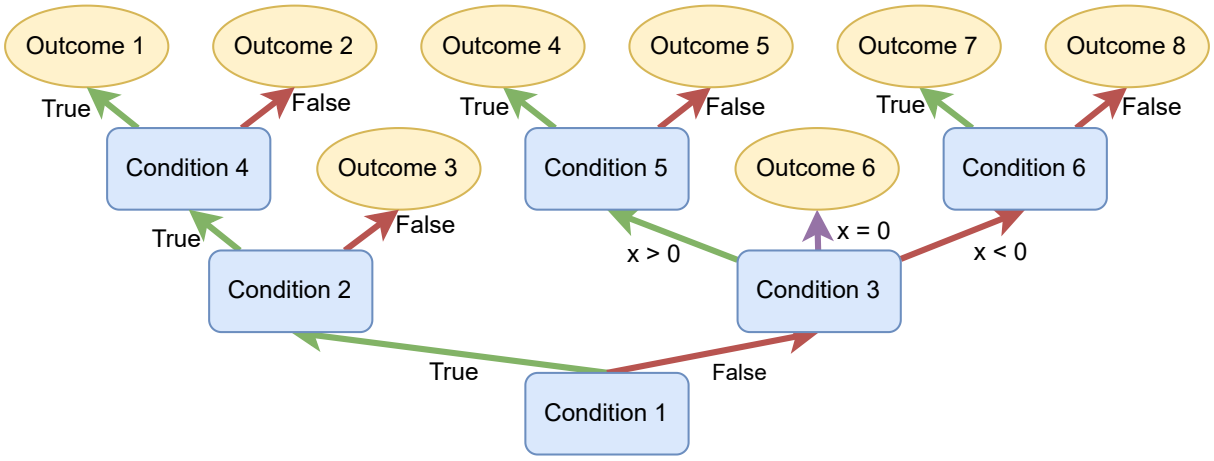


Figure 16: Decision tree example

In the Figure 16 example, there are eight possible outcomes where each outcome has only one possible combination of conditions. For instance, for outcome number four, condition one needs to be false, condition three requires variable x to be greater than zero and condition five needs to be true. Only with this condition/variable combination outcome four is achieved.

3.7 Graph

A graph is a structure with nodes or points tied together through connections, edges or links. It is mostly used in discrete mathematics and has multiple types of graphs [41]. A weighted graph is used throughout this dissertation, meaning every connection has a weight associated with it. These weights can have different meanings depending on the applied situation. Figure 17 shows two types of graphs, regular (a) and complete (b).

Figure 17: Graphs: (a) Regular; (b) Complete

The graph in Figure 17 (a) has five nodes and six connections and the graph in Figure 17 (b) has five nodes but with ten connections in between the nodes. Their mathematical notation is visible in Equation 3.6 for graph (a) and in Equation 3.7 in the graph (b) where N is the group of nodes and C is the group of connections.

$$\begin{cases} N = \{1, 2, 3, 4, 5\} \\ C = \{\{1, 2\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{3, 4\}, \{3, 5\}\} \end{cases} \tag{3.6}$$

$$\begin{cases} N = \{1, 2, 3, 4, 5\} \\ C = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\} \end{cases} \tag{3.7}$$

3.8 A* Path Finding Algorithm

A* is a path-finding algorithm for weighted graphs and is an efficient way to find the least cost path [42]. The weights of the connections between nodes define the cost of the path. The A* evaluates the path from the starting node to the ending node, trying to minimise the overall cost. In Equation 3.8 n is the next node, $g(n)$ is the cost of travel to the next node and $h(n)$ is a function that estimates the remaining path to the end node. A* iteratively evaluates the surrounding nodes and calculates their total cost $f(n)$. After calculating the values for all the surrounding nodes, it proceeds to that node and repeats the process.

$$f(n) = g(n) + h(n) \tag{3.8}$$

Figure 18 shows a step-by-step example where the cost of moving sideways is 10, and the cost of moving diagonally is 14. Step one evaluates two possible nodes and goes to the lowest cost node, reaching a dead end in step two. The overall cost $f(n)$ is in the centre of each node, the cost of travel $g(n)$ is in the lower left corner, and the remaining travel cost $h(n)$ is in the lower right corner.

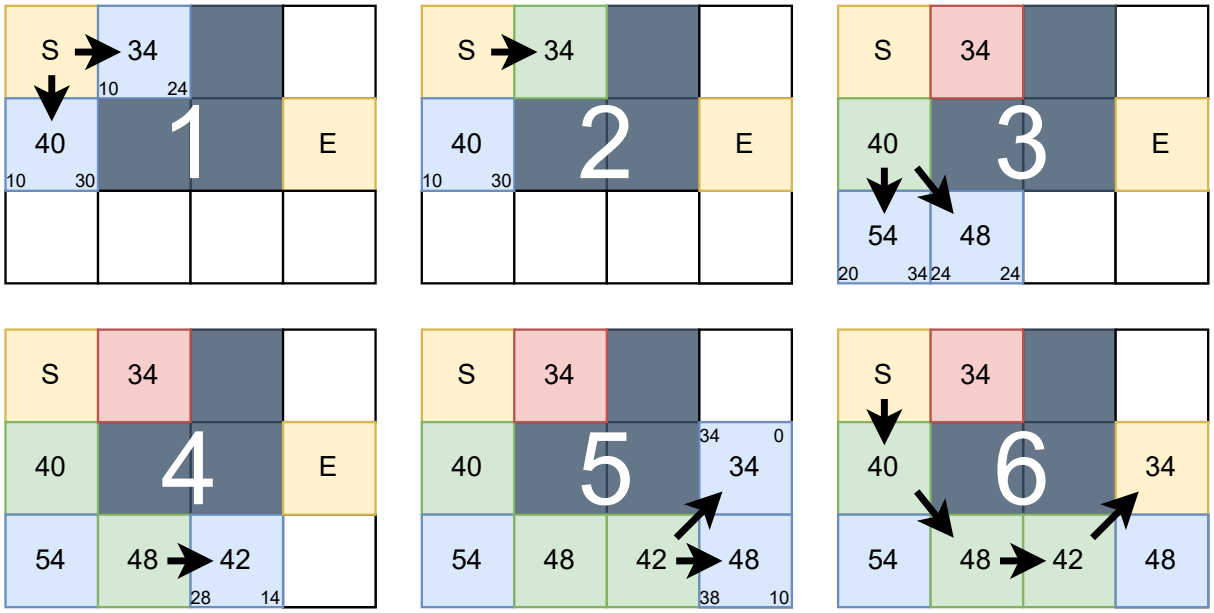


Figure 18: A* algorithm step-by-step example

3.9 Hungarian Algorithm

The Hungarian algorithm is an efficient technique for solving assignment problems. An assignment problem has the same number of elements and tasks; each element must be assigned to one task. Each element task assignment has a cost associated with it, and this method solves the problem through matrix manipulation, yielding the overall lowest cost possible.

The Hungarian algorithm uses a matrix where each row represents a task, each column an element, and each cell the cost of that element-task [43]. A series of steps for manipulating the matrix is carried out until there are enough zeros in the matrix. It is required to have lines passing through the zeros where the number of lines is the same or bigger than the number of tasks or elements. When this happens, a choice of zeros from different lines is a solution with the lowest cost overall. An example is shown in Figure 19.

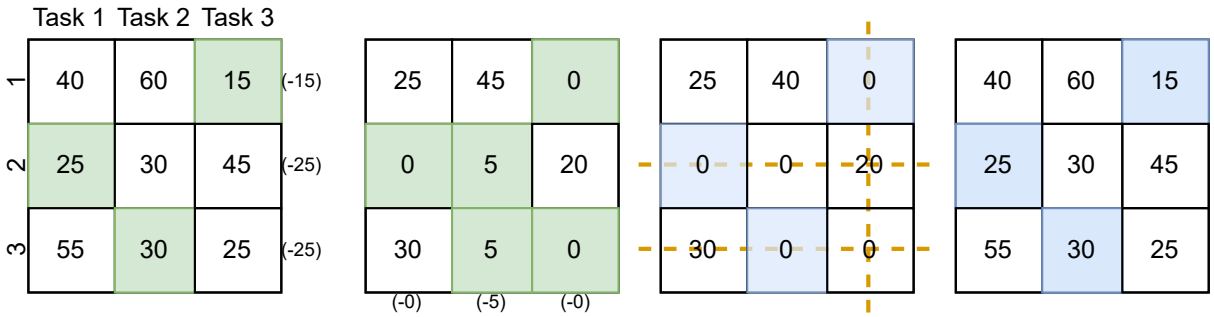


Figure 19: Hungarian algorithm example

The first step is to subtract the minimum value of each row from all values of the row. Next, the same must be done to the columns with the minimum value of each column. Lastly, if the number of zeros is equal to or greater than the number of rows, a combination of non-crossing zeros is the position of the solution. In this example, the solution was:

- Element 1 -> Task 3;
- Element 2 -> Task 1;
- Element 3 -> Task 2.

The overall cost of the solution presented is 70, which is the sum of the cost of each assignment, 15 from task three, 25 from task one and 30 from task 2.

Chapter 4

Robot's Hardware

The robot's hardware and mechanical description are presented to help better understand the development environment for the strategy. The hardware development occurred parallel to the strategy development and was crucial to achieving some dissertation objectives. All real-world results were acquired with these robots.

4.1 Architecture

Every robot is divided into different modules, each responsible for a specific task. The modules have the necessary sensors and actuators, but some sensors are also necessary in multiple parts of the robot. For example, the compass is present in the ESP32 module but influences both the control systems and the localisation system. Figure 20 shows a robot schematic with all its modules, how it interacts with the base station, and the protocols used between each model.

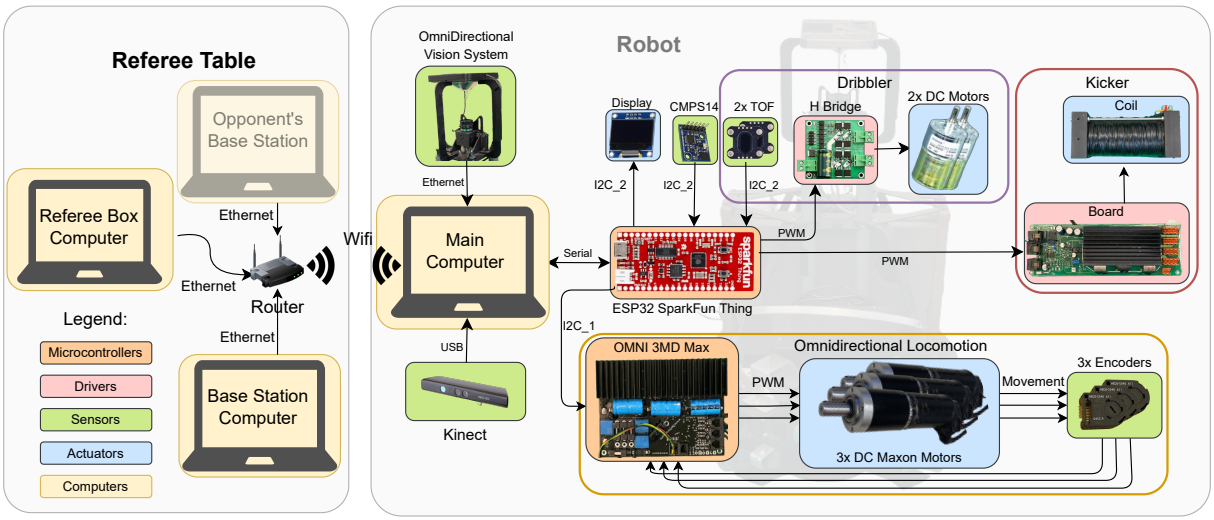


Figure 20: Robot's hardware diagram

4.1.1 Robot's Diagram

The LAR@MSL team has five robots: four strikers and a goalkeeper. All the strikers have the same hardware configuration and software, but the solution for the goalkeeper to optimise its objectives is slightly different. The differences between the strikers and the goalkeeper are the position of the Kinect Depth Camera [44], the upwards-extending arm, and the addition of a Light Detection and Ranging (LiDAR) sensor at the back of the goalkeeper. Figure 21 depicts a striker robot and its components.

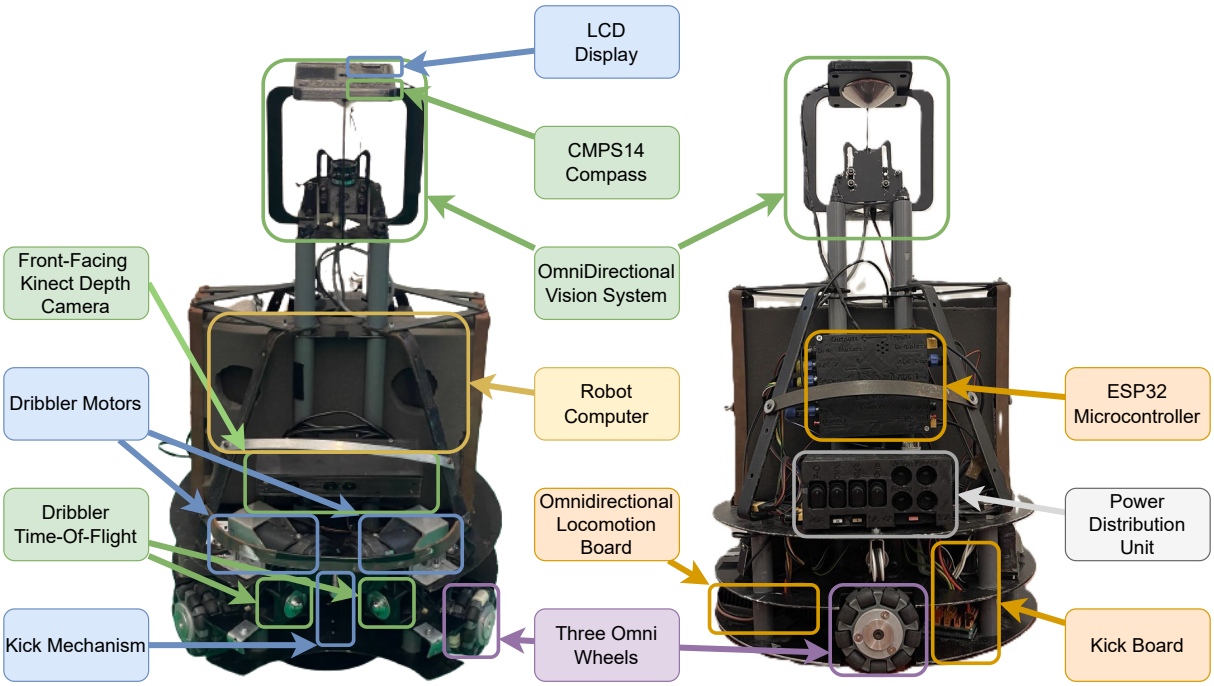


Figure 21: Robot's photo with the main elements highlighted (front and back views)

4.1.2 Chassis

Sometimes collisions can happen during a game, requiring a very robust structure, so most parts are made of iron or aluminium. Figure 22 shows the robot chassis and the aluminium boards that make up the layers of the robot. According to the size limits defined in the RoboCup MSL rule book [3], the robots cope with the dimensions (52 cm x 52 cm x 80 cm). The robot weighs 36 Kg, 4 Kg below the MSL rule book defined limit [3]. Even though a higher weight requires more energy consumption, having a lighter robot is worse when it comes to collisions and respective inertia. Besides all the metals around the robot, there is a protective cushioned fender. This is important so no sensible hardware is exposed. Should they collide with something or some other robot, the impact is absorbed. When playing with or against humans, it is also mandatory that the robots wear thicker skirts to protect the humans in case of shock.

(a) (b) (c)

Figure 22: Robot chassis photo: (a) and (b) Side view; (c) Aluminium plate

4.1.3 3D Printed Parts

As some custom-manufactured parts needed to have higher precision and due to the team's reduced budget, some parts were 3D printed either in Polylactic Acid (PLA) or Resin. Most of the 3D printed parts were boxes or fixtures, and Figure 23 shows some of them. Another example of 3D printed parts were some rollers for the omnidirectional wheels, but those were printed in resin due to the weight and impact.

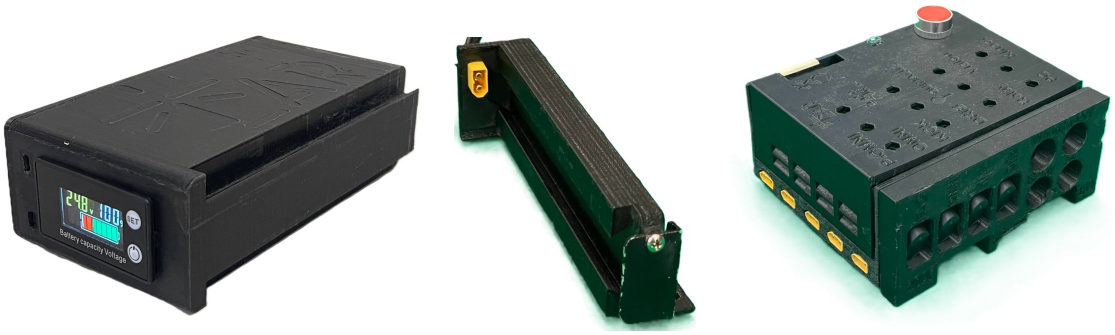
(a) (b) (c)

Figure 23: 3D printed parts and respective CAD[1] drawing: (a) Omni-vision camera support; (b) Ball caster wheel support; (c) Power box

[1] CAD (computer-aided design) software is used by engineers and designers to create 2D and 3D models of physical objects and systems.

4.2 Energy and Power Distribution

Regarding energy, the robot uses 24 V lithium-ion batteries with 6600 mAh. These were handmade by the team due to budget reasons. The construction is described in more detail in section Batteries in Appendix A - Extra Work. The batteries have a 3D-printed case and rail so that they are easy to connect, and the rail prevents them from being plugged backwards. Not all modules work at the same voltage, and hence the converters and switches are available in the Power Box. Figure 24 shows the batteries and Power Box. The robot has a remotely operated emergency stop mechanism, which consists of a two-channel wireless-operated relay on a low radio frequency of 433 MHz with a remote control. This relay does not cut power to the whole robot but turns off the Omni-3MD Max, the motor driver board Peripheral Interface Controller (PIC) so that the motors stop.



(a) (b) (c)

Figure 24: Energy management: (a) Lithium-ion 24 V battery; (b) Battery connection rail; (c) Power box

4.3 Low-Level Control and Sensors

The robot has an ESP32 Sparkfun Thing as the main microcontroller to manage communications among all hardware. It communicates with the main computer through a serial port, and most of the sensors use the Inter-Integrated Circuit (I2C) protocol. The ESP32 has two I2C channels being used for the board responsible for the robot movement, and all the other I2C devices are on the other channel. For safety purposes, the robot must stop as quickly as possible. A saturated I2C channel could pose an issue in achieving this objective. The devices on the second I2C channel (the compass, the Time Of Flight (TOF) sensors, and the display) are not a priority and, therefore, can be in a second channel. Figure 25 shows the Printed Circuit Board (PCB), the ESP32, the compass and the screen.

(a) (b) (c)

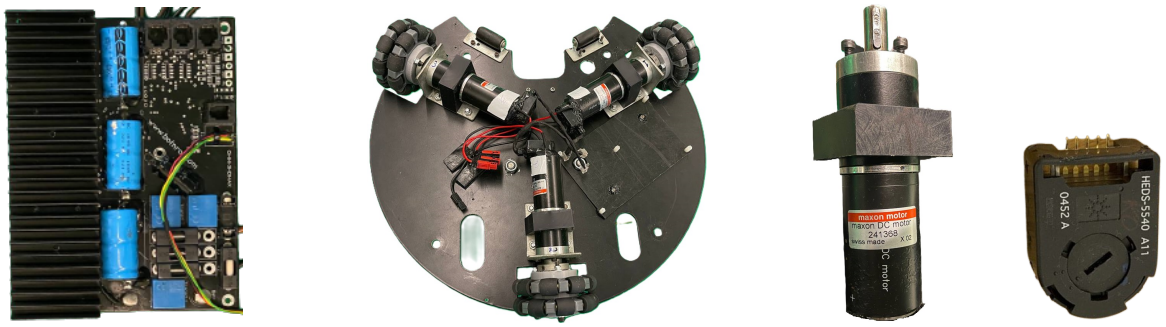
Figure 25: Low-level microcontroller components: (a) ESP32 and PCB; (b) ESP32 board box mounted on the robot and protection; (c) CMP14 compass and screen

4.4 Locomotion

Speed and agility are extremely important in football robots, and for that reason, a fast and efficient method of locomotion is important. The robots have three omnidirectional wheels at 120° of each other, allowing the robots to move in any direction at any moment while simultaneously rotating.

4.4.1 Omni-3MD Max

The motor driver board is the Omni-3MD Max, a board developed by Bot'nRoll, which is responsible for controlling the three motors based on the values given. The board receives a linear velocity, a direction and an angular velocity and calculates the kinematics necessary to reproduce that movement. It also handles the motor Proportional – Integral – Derivative (PID) control to have an accurate velocity, and even controls the acceleration curves to avoid drifting and wheel spin. Figure 26 shows the Omni-3MD Max board (a) as well as the motors attached to the chassis (b).



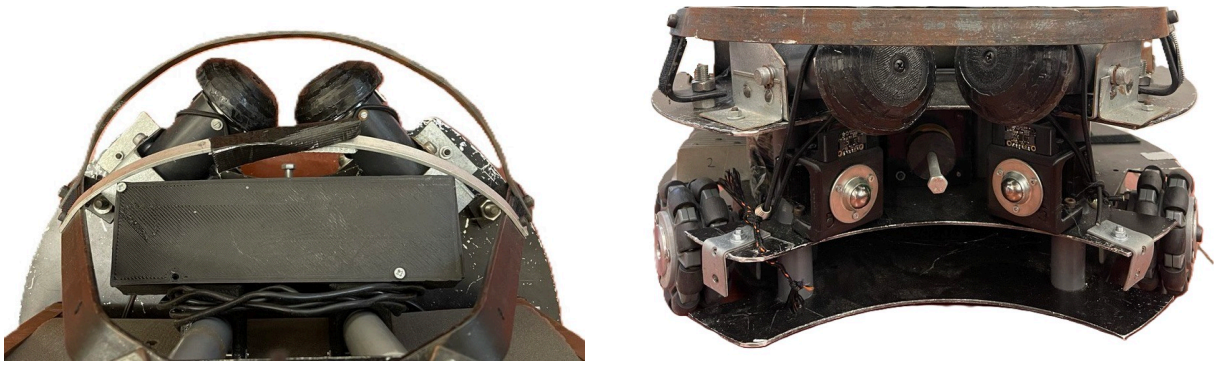
(a) (b) (c) (d)

Figure 26: Locomotion system: (a) Omni-3MD Max; (b) Motors and wheel; (c) Maxon motor; (d) Encoder

The motors are Maxon 148867 DC 24 V and have a maximum power output of 150 W each (Figure 26 (c)). Each motor has an encoder with the reference HEDS-5540, as shown in Figure 26 (d) and its output goes into the Omni-3MD Max for the control systems.

4.5 Ball Handling Mechanism

One of the robot's requirements was to be able to catch and control the ball. To solve that problem, each robot has a dribbling system able to receive and control the ball with two motors and two TOF sensors. The motors can move the ball in any direction, and they are controlled based on the two TOF distance sensors that are mounted facing the ball. The wheels were custom-made so that the ball's grip was maximised to pull the ball to the centre of the robot. Figure 27 shows the dribbling system as well as the TOF sensors and how they are mounted.

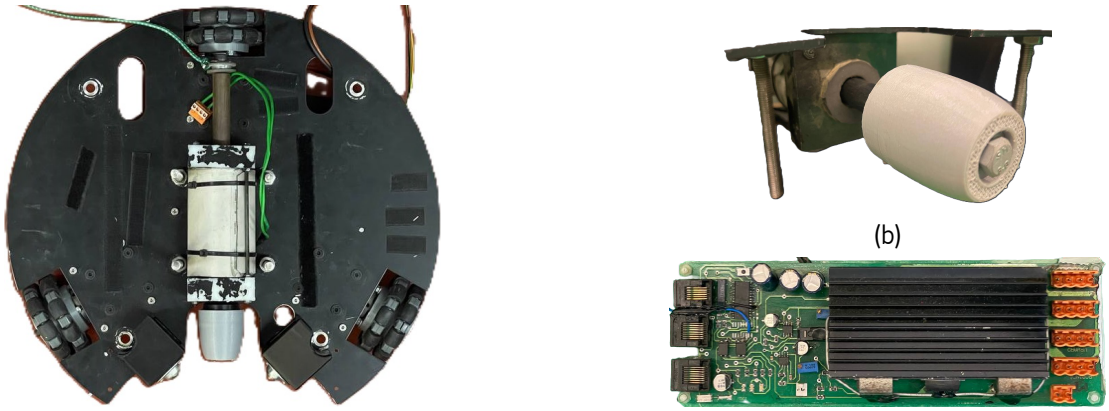


(a) (b)

Figure 27: Ball handling mechanism: (a) Top view; (b) Front view with the ball caster wheels

4.6 Kicking Mechanism

Another ability required by the robot is to kick the ball and for that, the robot has a kicking system. It is composed of two parts: the mechanical and the electronic parts. A coil and an iron flap are responsible for hitting the ball in the mechanical part. The electronics part is a board that charges capacitors to 450 V and controls the energy flow. When the kicking trigger is on, it sends the capacitor's charge into the coil that magnetically moves the iron flap and hits the ball. Work is being carried out to make the robot able to kick upwards in a projectile trajectory, and the importance of that is based on how difficult it is to defend an arc ball instead of a rolling ball. The kicking mechanism and board are visible in Figure 28.



(a) (b) (c)

Figure 28: Kicking mechanism: (a) Mounted on the robot; (b) Iron flap; (c) Kick board

4.7 High-Level Computation and Main Computer

Each robot has a laptop responsible for managing, calculating and controlling all high-level systems. It gets information from all the cameras and sensors, handles communications with the base station, runs all logic and control algorithms, makes decisions and even logs data. This central computer has a *Ryzen* 5 CPU with 8 GB of Random-Access Memory (RAM) and an *NVIDIA GTX 3060* Graphics Processing Unit (GPU). The significance of the graphics card lies in its ability to execute all the neural networks used in the detection algorithms. Regarding the operating system, it uses Linux, more specifically, the *Ubuntu* 22 distribution. Most of the code running on the robot computer was written in *Go* (a compiled language) [7].

4.8 Vision System

One of the most important robot input systems is the vision system, which consists of 2 cameras. The main camera, the catadioptric omni-vision camera, has a full 360-degree field of view. The second camera, a *Kinect* from the "Xbox 360", is a front-facing depth camera [7].

4.8.1 Catadioptric Vision System

The catadioptric vision system applied to the robots is an upright camera facing a conic/hyperbolic mirror, and its designation is the omni-vision camera [45]. The camera used is a Blackfly PGE 13S2C network camera that communicates through an ethernet cable using sockets.

On top of the camera, there is a curved mirror centred with the camera lens, making it able to see all around the robot. The angle of a specific object detected by the camera is given by the angle of that

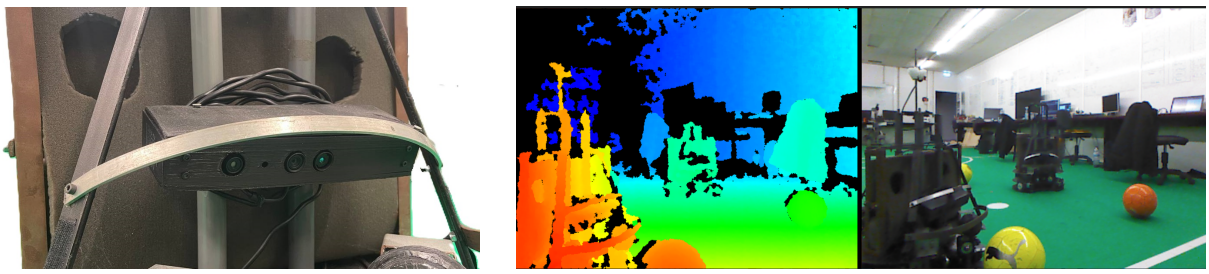
object to the centre of the mirror, and the distance is calculated based on the distance to the centre of the image. Figure 29 (a) shows the camera and the mirror, and Figure 29 (b) shows an image taken from the camera in a game situation.

(a) (b)

Figure 29: Omni vision camera system: (a) Camera mount and mirror; (b) Image from camera

4.8.2 Kinect Depth Camera

The second camera on the robot is a front-facing depth camera, a *Kinect* from the "Xbox 360". It is an infrared camera that is able to capture Red-Green-Blue-Depth (RGBD) images at 30 frames per second. This front-facing camera is used for most of the ball actions such as kicking, receiving, and attacking the ball. The need for the second camera is due to having more accurate detection systems and, consequently, better control systems. The downside of the *Kinect* camera is its low field of view, so when the *Kinect* is not able to see the ball, the information used is only from the omni-vision camera. Figure 30 shows the *Kinect* camera and how it is mounted on the robot.



(a) (b)

Figure 30: Front-facing depth camera Kinect: (a) Camera mount; (b) Image from camera depth and colour

Chapter 5

Communication System

The communication system is an indispensable part of a centralised strategy and robot synchronism. It enables the robots to communicate via a Wi-Fi network and all the communication limitations and restrictions were coped with, based on the MSL rule book [3].

5.1 Communication's System Description

The robots use UDP sockets to communicate with the base station which is the team's central computer. The communication system is the same in the simulator and the physical robots to help transition between the simulator and the real-world robots. Robots can communicate with each other if all packets go through the game's access point, but the LAR@MSL team does not use that feature. Also, in the same network is the RefBox computer, which is responsible for translating the referee instructions to the team's base stations and, consequently, the team's robots [9]. The communications diagram is shown in Figure 31.

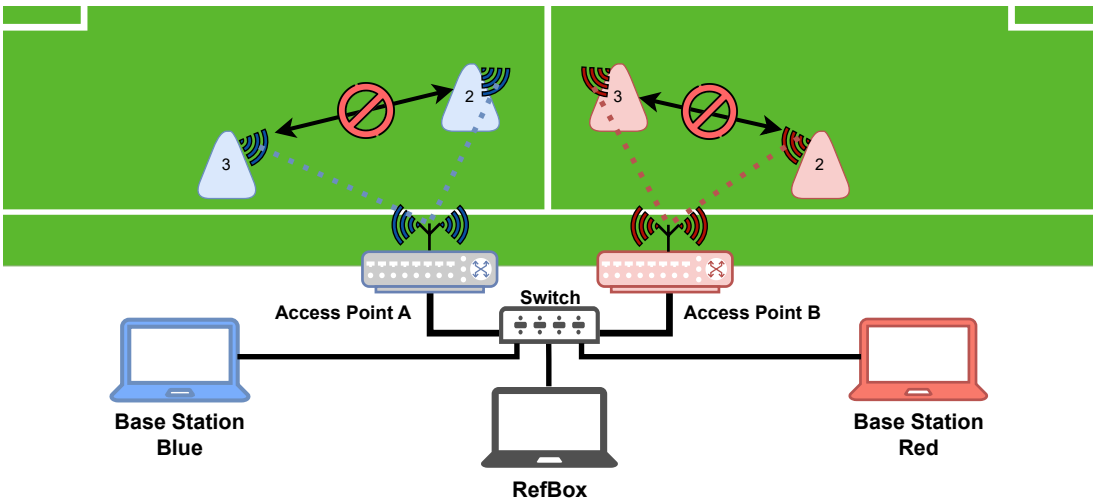


Figure 31: Communication system diagram

5.1.1 Competition Environment

In competitions such as the "Festival Nacional de Robótica" and RoboCup, the Wi-Fi environment is extremely saturated. This saturation is due to the fact that Wi-Fi networks have little or no organised allocation of frequency for respective leagues. Even though the MSL has dedicated routers, there is no control over the surrounding Wi-Fi environment. This is also the reason for the choice of UDP sockets over TCP. The communication system's speed is crucial to ensure all decisions are occurring in real-time and not with a delay. Also, since the desired information is always the most recent, acknowledging a previous TCP communication is not at all desirable. In real-time, fast-moving robots losing one packet of information is preferable to working with outdated information.

The MSL league has access points operating on both the 2.4 GHz and 5 GHz frequencies. However, these frequencies are heavily saturated due to the presence of other leagues, television channels, personal hotspots, spectators and various other sources. An example of the network environment acquired in RoboCup 2023 is shown in Figure 32. Each arc present in the graph is a different network.

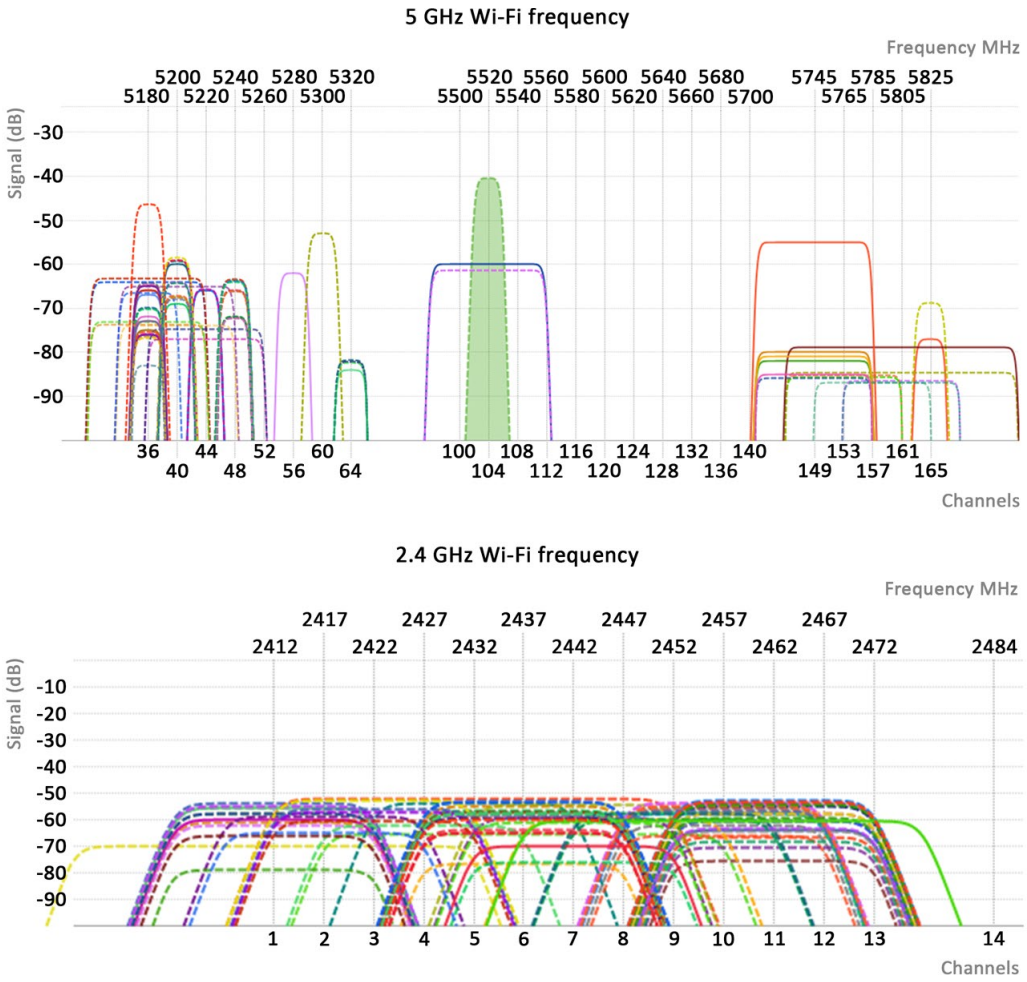


Figure 32: Saturated environment in RoboCup 2023 (5 GHz and 2.4 GHz)

5.1.2 Bandwidth Limitation

The rule book [3] specifies a bandwidth limitation so that no team saturates the Wi-Fi, limiting the opponent team's bandwidth. All teams have the same network limits, whichever Institute of Electrical and Electronics Engineers (IEEE) 802.11 mode is being used. The slower mode (IEEE 802.11b specification) is the one that actually limits the amount of data that can be transmitted. Each team is then allowed to use at most 20% of the bandwidth. The maximum bit rate that can be used by any team is 2.2 Megabits/second.

5.2 Organisation of IP Addresses and Ports

Each team has a predefined range of Internet Protocol (IP) addresses and a subnet mask to work with, specified in the MSL rule book [3]. Each team has access to tables with the list of IP masks for both multi and unicast communications. The mask of the LAR@MSL team for unicast IP is 172.16.49.* and for multicast, the IP address is 224.16.32.49. Currently, the team only uses the unicast IP.

5.2.1 IP Addresses List

Table 1 shows a list of all the robots and all the team computer IP addresses. The base station and backup are still under the team mask, but the RefBox computer is shared between all the teams.

Table 1: Table of robots and base station IP.

Robot	IP	Computer	IP
Robot 1	172.16.49.11	Base Station	172.16.49.16
Robot 2	172.16.49.12	Base Station Alternative	172.16.49.17
Robot 3	172.16.49.13	RefBox	172.16.1.2
Robot 4	172.16.49.14		
Robot 5	172.16.49.15		

5.2.2 Port Organisation

Each robot has three UDP sockets being two sockets used unidirectionally that communicate at a high frequency and one used bidirectionally for debugging purposes. When calibrating the control systems, the debugging sockets are mostly used to send and receive PID parameters. Information is only sent when a change in the parameters occurs either in the robot or in the base station, and therefore, these sockets are not used during the game. The port organisation is shown in Figure 33 where the high-frequency sockets are in black, and the debugging sockets are in blue. The communication with the RefBox is also

made through a socket and the used ports are also available in Figure 33. In the simulator, two additional modules are present that communicate with the robots, the main supervisor and the field display. These modules and their purpose are explained in section 6.3 Architecture, Controllers and Supervisors in the Simulator chapter.

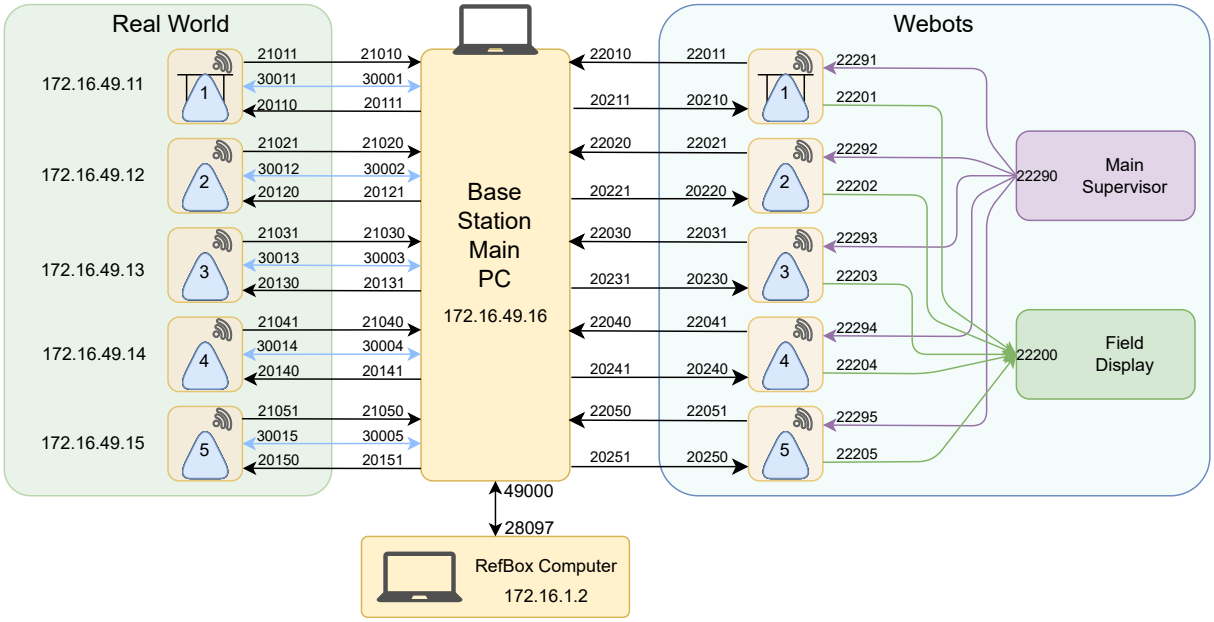


Figure 33: Sockets organisation diagram for real-world and simulation environments

5.3 Data Frames

In order to play the game, robots need to gather all the information required by the strategy or robot actions. Only the data collected by robots can be used by the teams. No external sensors can be used. For this reason, customised packets have been created to transmit and receive all the essential information, such as sensor readings, motor commands and status alerts. The communication system is crucial in consolidating the data obtained from all robots so that it can be used reliably by the strategy.

5.3.1 Information Sent to the Base Station

The information sent by the robots consists of two big groups, game information gathered by the robot and the robot information. The game information is, for example, the ball detection, the robot's localisation, etc. The robot information is about the robot itself, its behaviour, sensors, states and variables. The communication frequency of this data sent by the robots to the base station is about twenty-five times per second.

Strategy decisions are taken based on the information generated after a data fusion occurs between all the robot's information. Table 2 shows all the game information gathered by the robot and sent to the base station. The ball information is gathered by the omni-vision camera as well as the front-facing Kinect camera. The ball detection on the images is carried out with a neural network. The ball information is its X and Y coordinates, angle, distance and confidence value. This is relative to the robot's location and the confidence value given by the neural network. The information regarding ball possession on its dribbler system is given out by the handler variable.

The localisation information provides the base station X and Y coordinates of the robot on the field and the robot's orientation, which is calculated by the omni-vision camera and the CMPS14 compass. The omni-vision camera also detects all robots and humans on the field within a 15-meter distance and separates teammates from opponents based on colour.

Table 2: Game information sent by the robots

Variable	Type	Content	Description
Ball	List[float]	[X, Y, Ang, Dist, Conf, Handler]	Ball information
Localisation	List[float]	[X, Y, Orientation]	Self localisation
Teammates	List[List[float]]	R1_X R1_Y R1_Ang R1_Dist R2_X R2_Y R2_Ang R2_Dist R3_X R3_Y R3_Ang R3_Dist R4_X R4_Y R4_Ang R4_Dist	Teammates, robots and humans
Opponents	List[List[float]]	R1_X R1_Y R1_Ang R1_Dist R2_X R2_Y R2_Ang R2_Dist R3_X R3_Y R3_Ang R3_Dist R4_X R4_Y R4_Ang R4_Dist (...)	Non-team elements robots and humans

Table 3 shows all the robot information that does not affect the game directly but is important for managing, debugging and making sure all the robot's hardware is functioning correctly. It shows battery power percentages, communication quality variables, CMPS14 compass values, dribbler and directional motor values, temperatures, CPU and GPU usage and temperatures, as well as loop times and much more information. All this information is displayed in real-time in the base station and is updated with the same frequency as the remaining game values.

Table 3: Diagnostics information

Variable	Type	Range	Description
battery	float	$0 - 100\%$	Robot battery percentage
QC1	float	$0 - 100\%$	Communication quality between ESP32 and PC
Cap	int	$0 - 100\%$	Kick capacitors charge
Compass_bearing	float	$0 - 360°$	CMPS14 bearing value
linear_vel	int	$0 - 100$	Linear velocity applied to the motors
angular_vel	int	$0 - 100$	Angular velocity applied to the motors
direction	float	$0 - 360°$	Movement direction
vel_db_r	float	$0 - 100$	Right dribbler motor velocity
vel_db_l	float	$0 - 100$	Left dribbler motor velocity
disp_db_r	float	$0 - 500mm$	Right dribbler laser distance measured
disp_db_l	float	$0 - 500mm$	Left dribbler laser distance measured
OMNI_temp	float	$0 - 100°C$	Omni-3MD Max temperature
buttons	int	$0 - 7$	ESP32 button state
CPU	float	$0 - 100\%$	PC CPU usage
cpuTemp	float	$0 - 100°C$	PC CPU temperature
GPU	float	$0 - 100\%$	PC GPU usage
gpuTemp	float	$0 - 100°C$	PC GPU temperature
pcbattery	int	$0 - 100\%$	PC battery percentage
dt	int	$0 - 9999ms$	Main code loop time
commit	string	$0000000 - fffffff$	Current main code commit
ID	int	$0 - 128$	Communication ID to control the flow
dtComms	float	$0 - 9999ms$	Time between packets received by the robot

5.3.2 Information Sent to the Robots

The robots are fully autonomous and gather all the information available, but decisions are taken in a central computer and to assign skills, information needs to be given to the robots. The packets sent to the robots are composed of the skill and the respective skill arguments. More details about the skills, their purpose and how they are attributed to the robots are given in the section 7.3 Skills.

Chapter 6

Simulator

In order to accelerate the development process and to allow the team to work on both the physical robots and game strategy at the same time, a simulation tool was created. This tool was specifically designed for game strategy which facilitates development and does not require any robot hardware, sensors or control systems. It was also developed in a way to facilitate the transition between the real-world environment and the simulated one.

6.1 Software Used

On developing a simulation environment, there were two big options available: creating a simulation environment from scratch or using existing simulation software like Webots, Gazebo or CoppeliaSim. As creating a simulation tool from scratch has a few upsides due to time constraints, and since it was not the core of the dissertation, it was decided to develop a simulation environment based on one of the most common software used. For strategy, a 2D simulation environment would be enough, but since it is an objective to make the robots kick in a projectile trajectory, a 3D environment was required. This way, the simulator tool could simulate not only the arc kick but the goalkeeper extending arms as well.

Learning Webots was not as simple as other simulation software like CoppeliaSim, but its documentation and tutorials were accessible and complete. It has compatibility with various programming languages, and the API to control the environment was intuitive and powerful. The transition between the real-world and the simulator needed to be kept simple, so the communication system between the base station code and the simulator is the same as in the real-world scenario, all based on sockets. There were other alternatives regarding communications with external controllers, although performance and simplicity of the transition were important factors.

6.2 Components, Field and Organisation

The simulated environment was created with the focus of accelerating the game strategy development, so the environment was kept as simple as possible (Figure 34). Since there was no need to have the omni-wheel physics, the dribbling system and other moving parts were developed with simpler solutions so that the simulator would have better performance.

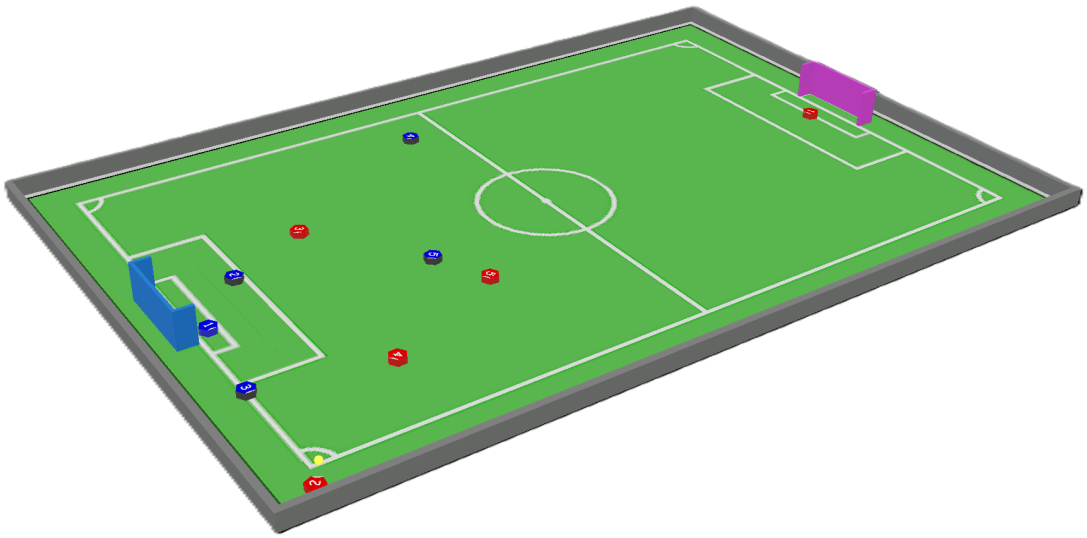


Figure 34: Webots field, simulation elements and robots

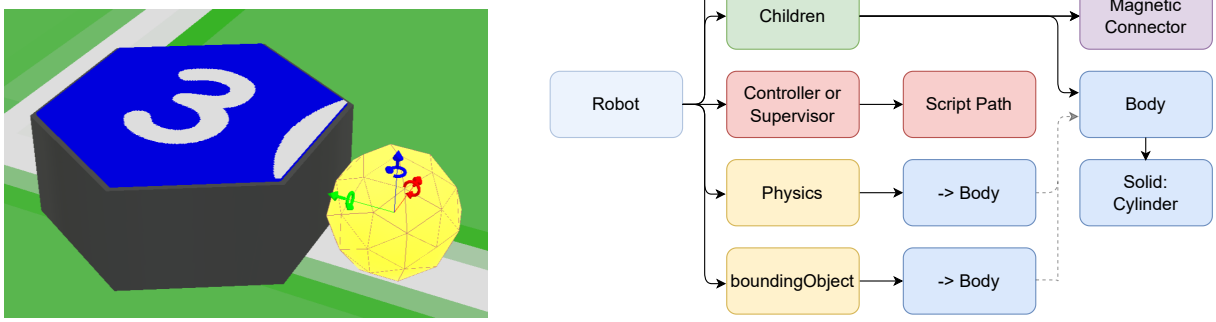
The simulated environment has ten robots, two goals, a ball, and other extra components to help better optimise and manage the simulation. Besides the robots and the ball, no other objects move, as they still need to be able to be interpreted by the physics engine collision system. The elements that still detect collisions and have no movement are called static objects.

6.2.1 Static Simulation Objects

The goals, the field floor and the walls are solid-state static elements present in the simulation. The only physics calculations carried out with these elements are the collision system. These elements do not move but still behave accordingly when an object collides with it. The ball is neither a robot node nor a static object, but it has gravity applied to it and also has an associated weight and elasticity so that the ball's behaviour is similar to the real world. The robot and ball use controllable magnets to simulate the dribbling system. The ball has a passive magnet, while the robots can activate or deactivate their magnet to catch and release the ball respectively as they would in the real world with the dribbler mechanism [46].

6.2.2 Single Robot

In the simulation, each robot is an extruded hexagonal shape and has a magnetic actuator in the front, as shown in Figure 35 (a). The robot does not have wheels because dedicating computational power with omni wheels, PID control systems or kinematics would be inefficient. Instead, the robot's movement is controlled by applying a movement vector to the robot itself with the supervisor API, with the arguments: linear velocity, direction, and angular velocity, similar to the real world. The dribbling mechanism uses magnetic actuators to either grab or release the ball. When the ball is close to the robot's front, the magnetic actuator is activated, and the ball remains attached to the robot until it kicks or releases the ball by deactivating the magnet. The kicking mechanism is implemented by code and not a simulated actuator. When the robot kicks, it applies force to the ball, pushing it away. The kick force also varies based on one of the kicking parameters. The robot's nodes hierarchy is shown in Figure 35 (b).



(a) (b)

Figure 35: Simulation robot element: (a) Body and texture; (b) Node hierarchy

6.2.3 Opponents

The opponent robots in the simulator are shaped like extruded hexagons with a red colour and can be moved manually. They are identical to the team robots in every aspect and can be controlled by the *control* skill through the base station. They also have a magnetic actuator and can kick just like the team robots. This allows the testing of strategies in multiple ways such as static plays, both teams playing with the same strategy and the strategy against humans. In this last case, the game can be played with five human players controlling the opponents using game controllers. Each human player controls one opponent, playing against the developed strategy to ease the game strategy calibration. This situation is visible in the Tests and Results chapter, section 9.5 Against Humans.

6.3 Architecture, Controllers and Supervisors

Webots can have multiple codes running simultaneously, and there are two types of scripts, the controllers and the supervisors. Controllers and supervisors are processes, explained in section 6.3.3 Controller vs Supervisor with the purpose of gathering information, actuating in the simulation environment, and can even do both. They are either asynchronous or synchronous with the simulation time step. Every interaction between a controller or a supervisor with the simulation environment is carried out through the Webots API. These functions can be time-consuming and so reducing the number of requests is a requirement to optimise it.

6.3.1 Architecture

All supervisors and controllers in the simulation run in parallel, so communication between them is a requirement. In this case, even though Webots have communication tools developed to link two or more controllers, sockets were the solution opted for in order to use the same method of communication as in the real world. Another reason for this choice was the reusability of code should the simulation software ever change, or even in the event of over-the-internet simulations. Figure 36 shows the simulation architecture, where the yellow arrows are the sockets used to send information from the base station to a robot, either from the own team or a simulation opponent. The blue arrows are the sockets used by the robots to send information to the base station or the display controller. The purple arrows are the sockets from where the main supervisor sends world information to the robots. The dashed red arrows are API functions either to send commands or receive information.

The exact port numbers used by the team robots with either the base station, main supervisor or display controller are available in Communication System in section 5.2.2 Port Organisation.

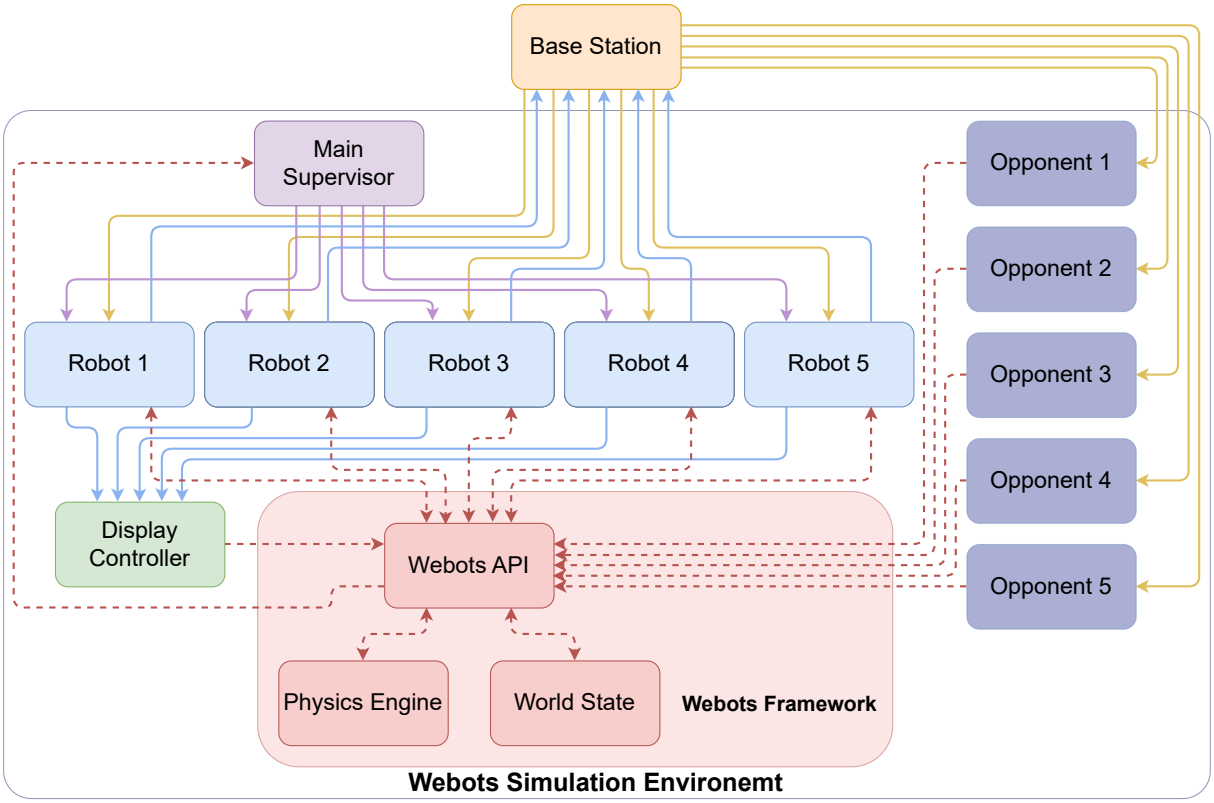


Figure 36: Webots simulation architecture

6.3.2 Nodes and Webots API

Nodes within Webots are the fundamental building blocks of the simulation environment. They represent entities such as robots, objects, sensors, actuators, lights, or any other element present. A simulated environment is organised in a hierarchical structure, where each node has specific attributes and functionalities. The organisation follows a parent-child relationship, where nodes can have parent nodes and children nodes. For instance, a robot node may have children nodes representing its sensors, actuators, or even subparts of the robot itself.

These nodes are manipulated and interact with the available Webots API. The API enables precise control of the simulation and allows code to create, modify, and interact with every node in the simulation. To use the API which is available in multiple programming languages, the node robot can have code that is either a controller or a supervisor. All controllers and supervisors require a parent robot node to interact and run in the simulation [46].

6.3.3 Controller vs Supervisor

In Webots, a controller and a supervisor are both used and they serve different purposes within the same simulation environment. A controller is a program with the focus of controlling a specific robot or a group of robots within the simulation. The controller only has access to children nodes inside its own, and it interacts with movements, sensor readings, and decision-making processes of the robot(s).

Supervisors, on the other hand, are higher-level controllers with a focus on managing the overall simulation environment rather than controlling individual robots or nodes. Supervisors can control the entire simulation, including the setup, initialisation, coordination of multiple robots, global environment settings, and interaction with external nodes. Supervisors are usually implemented using the Webots controller API, allowing them to access and manipulate the simulation environment. They are distinct from the controllers responsible for the robots' actions.

6.3.4 Main Supervisor

A main supervisor was created to reduce the number of API requests as well as synchronise the information between all robots. All the robots need to detect objects on the field. Instead of each robot requesting the location of ten robots and the ball, the main supervisor gets information about everything and gives that information to all robots. This method is less CPU intensive as well as faster and synchronous. The information is sent through a socket which is the same method for all the other inter-controller communications. This method reduces the number of API requests from 110 to 11 per simulation iteration, which is an improvement of ten times.

6.3.5 Display

To simplify the debugging process, visual representations are used both on the base station and in the simulator. A display was used as a replacement for the field floor within the simulation environment. This floor serves as a display, synchronised with a controller, facilitating a visual representation of the robot's active skills. The display has a resolution of 240×160 pixels where each pixel corresponds to a 10 cm distance in the real world. The purpose of this display primarily revolves around showcasing the active skills of the robots, including ball interactions, movements, and other skills. To effectively manage the field floor, a centralised controller has been developed. Each robot supervisor can make requests to the display controller to draw onto the field floor. Upon receiving these commands from multiple robots, the display interprets and executes the requested actions, resulting in the visualisation of the intended representations.

The display node on Webots has multiple API functions, with the ability to draw circles, ellipses, lines, and other graphical elements. All the display API functions are available to the robot's supervisors and only a few are used to represent the skills.

6.3.6 Robot Supervisor

In the simulated environment, each robot is equipped with its dedicated supervisor, ensuring individualised control and management for distinct robot nodes. Despite sharing the same base code, each robot operates through its supervisor instance, where the sockets, which are different in every robot, are used for inter-process communication among the nodes. These robot supervisors receive inputs from both the main supervisor and the base station.

After gathering all the information available, the robot calculates the desired movement and action for the active skill given by the strategy. The robot movements and actions are then sent to the Webots API to move the robot accordingly, dribble and kick if necessary. Information is also sent to the base station to serve as feedback to the game strategy algorithms, just like it would happen in the real world. Apart from the necessary communications for movement and feedback, the desired requests are made to the display controller to represent the robot's actions on the field floor. The data flow between all the controllers and supervisors is represented in Figure 37.

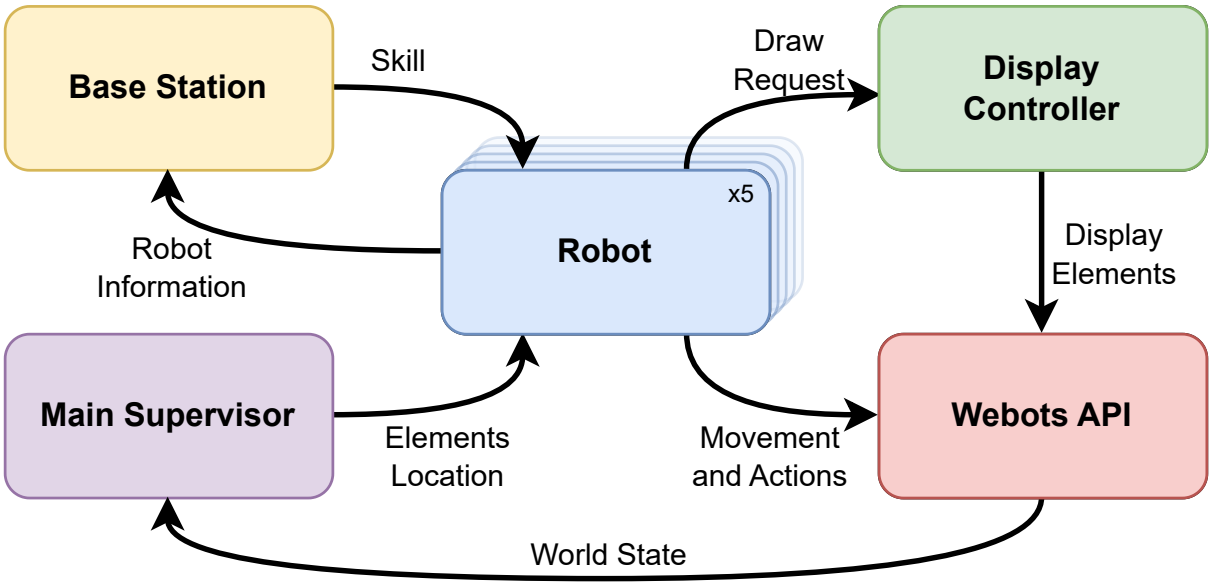


Figure 37: Webots simulation data flow diagram

Chapter 7

Game Strategy

RoboCup middle-size league teams have traditionally used game strategies where robots make autonomous decisions [3]. This dissertation introduces a new game strategy approach that relies on centralisation, probability-based plays, and fast communication, redefining teamwork among robotic football players. Unlike traditional methods, this approach retains robot autonomy while taking advantage of a central computer, the base station, to provide skills, world information, and even ideal paths. This centralisation enhances probability-based decision-making and synchronises robot actions, creating an efficient and reliable game strategy without a playbook.

7.1 Architecture

Figure 38 represents the colour-coded game strategy architecture schematic. It contains the Communications module, Data Fusion, Play Generator, Decision Trees [39, 40] and Heat Maps [38] and the Skill Assigner. The modules are explained in the following sections.

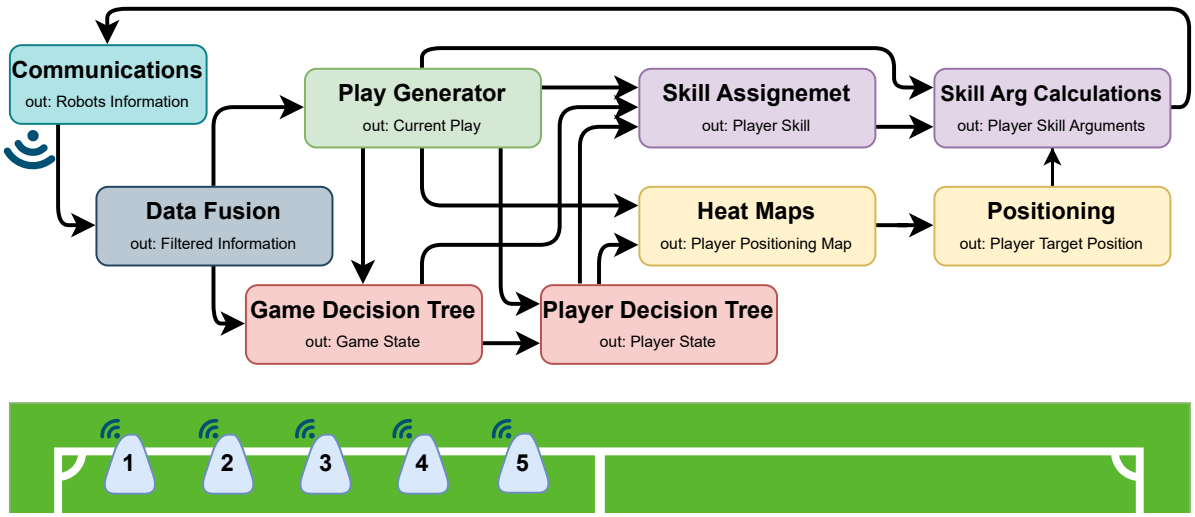


Figure 38: Probability-Based Strategy (PBS) architecture diagram

The communications module (in light blue) was already described in the Communication System chapter. 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 preview of the heat maps and positioning modules (indicated in yellow). The two modules in purple are tasked with assembling the information, calculating the necessary arguments for packets, and transmitting them to the communication modules [9], thereby ensuring synchrony among the robots.

7.2 Visual Representation

All base station representations are explained in chapter 8, but as context for the game strategy, a few key elements are presented. Figure 39 shows a MSL field, and the predefined axes X and Y used throughout the strategy. All the field locations and positions use this axes configuration. The team goalkeeper defends the negative X axis goal.

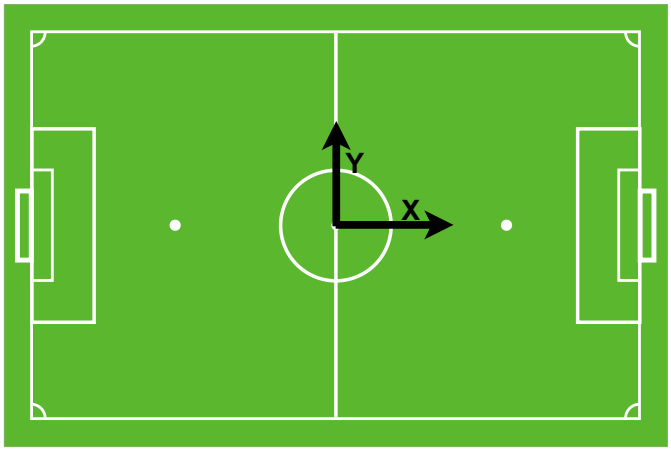


Figure 39: Field axes and size reference

Figure 40 shows the base elements. Represented with a blue hexagon are the team robots with their robot number (Figure 40 (a)). The arrow in front of each robot shows its orientation. The red circle represents the opponent's robots (Figure 40 (b)) and the small yellow circle represents the ball, visible in Figure 40 (c) next to robot number two. Between two teammates, one with ball possession, there is a yellow line that represents the line of pass, visible in Figure 40 (c). The perpendicular yellow line between an opponent and the line of pass shows the shortest path between the opponent and the line of pass, visible in Figure 40 (c), and its intersection is represented by a tiny yellow circle.

(a) (b) (c)

Figure 40: Representation guide: (a) Team robot; (b) Opponent; (c) Line of pass and opponent's influence

7.3 Skills

As the basis for a game strategy, the robots can perform certain skills as 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 game strategy. Specifically, in the LAR@MSL game strategy [7] and in this dissertation, each robot can be assigned one of eight possible skills, each with up to seven arguments (Table 4). The number of arguments is higher than necessary at the moment since it was planned to provide future expansion of the system.

Table 4: Table of skills and respective 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	-	-	-

Stop

Stop is the most basic skill of them all and has the purpose of stopping all robot-moving elements. It brakes the three locomotion motors, completely stops the dribbling mechanism and blocks the robot from actuating its kicking mechanism. It has no arguments and it is used every time the game is stopped or in case of an emergency. This skill has no hardware override and is not connected to the emergency stop mechanism since it only uses software. For an emergency stop, the robot has a wireless operated relay described in section 4.2 Energy and Power Distribution.

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, a flag to define if the robot's orientation is free or locked, and the X and Y coordinates the robot should orientate to. If its orientation parameter flag is free, by default the robot orientates itself to the ball location. It is one of the most used skills and requires information from the localisation and detection systems. The robot dynamically calculates its path so that it can avoid all obstacles while performing the desired movement.

Attack

The **Attack** skill is responsible for trying to grab the ball as fast as possible with the dribbling mechanism. It receives the ball coordinates as a parameter and only uses it if it does not see the ball with its own vision system. It gives priority to the front-facing Kinect camera, followed by the omni-vision system. Only if both are not able to see the ball, the robot will use the parameter sent with the command. Regarding the grabbing action and the ball control, it is entirely achieved by the dribbling mechanism. The P parameter is a flag used for penalties, in which case the robot moves slower towards the ball so that it does not take the ball out of place.

Kick

When the robot intends to score a goal or pass the ball, the **Kick** skill is used. The robot already knows its location by the localisation system. The first two parameters indicate the absolute position where the ball is supposed to be kicked. The third argument separates a pass from a kick to the goal. The robot adjusts the force applied to the kick accordingly. In a pass, the parameter specifies the kick force based on the distance to the teammate. When flagged to kick to the goal, the robot calculates where to shoot and kicks to the goal with maximum power.

Before kicking, the robot can move up to 3 meters with the ball. Where to move is calculated by the strategy. The two final parameters, D_x and D_y, are the ideal relative movement the robot should make before kicking. Before the kick, the robot might need to rotate itself to the desired position. Taking advantage of its omnidirectional movement, the robot can move to a location within a 3-meter radius in order to optimise and simplify the pass or kick to the goal. That position is calculated by the base station and takes into consideration the entire game and world state.

Receive

In pair with the kick skill there is the **Receive** skill, which instructs a robot to receive the ball coming towards it. It needs to move and orientate itself accordingly to precisely receive the ball and grab it with the dribbling mechanism. The only variable needed for a robot to receive a ball is to know where it is so that is the only parameter sent. This parameter is used just for the case when the robot does not see the ball.

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 such as the opponent team robots. For example, to prevent a pass, the robot can be instructed to cover point A from point B, and it calculates the desired coordinates based on those two arguments. As parameters, besides the two targets X_1, Y_1, X_2 and Y_2, the aggressivity of the player is also sent as a percentage. This consists of how much pressure a robot should put on one target or the other. Translated to the real world, that percentage defines where on the line between the two points the robot should position itself. The robot orientation is always to the second target with X_2 and Y_2 coordinates.

Defend

The **Defend** skill is only used by the Goalkeeper and this robot never leaves this skill. It is responsible for defending the goal and predicting the ball's movement and arc trajectory by positioning the goalkeeper. It also controls the arms of this robot. It receives just two parameters consisting of the ball location, and 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. The goalkeeper autonomously detects if a kick occurred in the goal direction, calculates the ball's trajectory and moves itself to defend the goal. It also actuates the goalkeeper's arms within the time and size restrictions defined in the MSL rule book.

Control

In demonstrations and test/debug situations (never during a game), this skill is used to control the robot remotely by the user. The robot is controlled by a gamepad controller connected to the base station computer or with this computer's keyboard. This skill is also used in the simulation tool since it controls the opponents for testing purposes. The parameters used are the movement in X and Y on the field, the desired orientation angle and the pass and kick parameter for the robot to kick.

7.4 Data Fusion

Information from the game can only be gathered by the robots on the field. No external data can be used, and so it is based on that information that game strategy and plays are generated and given to the robots. Robots send information to the base station at a rate of 25 Hz and its fusion is essential for the stability and accuracy of the game strategy and robots behaviour.

The data fusion is carried out every loop iteration. The first and most important data is the ball location. Using the distance from the robot to the ball and the confidence of the vision detection network, all robot's ball detection is fused to a weighted average. It is also based on this fused data that calculations for the ball movement prediction are carried out. These predictions are used to detect ball interceptions from opponents. It is also detectable when a robot is not seeing the ball correctly, and so in the skill arguments, the correct ball location is sent. The robot's location, especially the opponent robot, is also worked in order to have a better opponent location accuracy.

7.5 Play Generator

The basis of any football strategy is to decide the right play at the right time and having a synchronised team. Most teams have the decision-making process occurring in every robot and its synchronisation occurs afterwards. Another common aspect of other strategy algorithms is having a playbook, a predefined set of plays, and robots try to apply a known play to the current game situation.

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 adjustable tuning to various opponent team styles, including speed-focused, precision-oriented, or hybrid approaches. The risk tolerance and

safety measures [47] 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 specialised knowledge. Additionally, the algorithm leverages the homogeneity of the robots' capabilities since 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.

7.5.1 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 can be modified later for a real-world transition, if necessary. The distance between two robots, given their location, is first calculated as seen in Equation (7.1). Then, the probability of a successful pass without any opponent's influence is calculated in Equation (7.2). The meaning of the variables is presented in Table 5.

Table 5: Description of variables

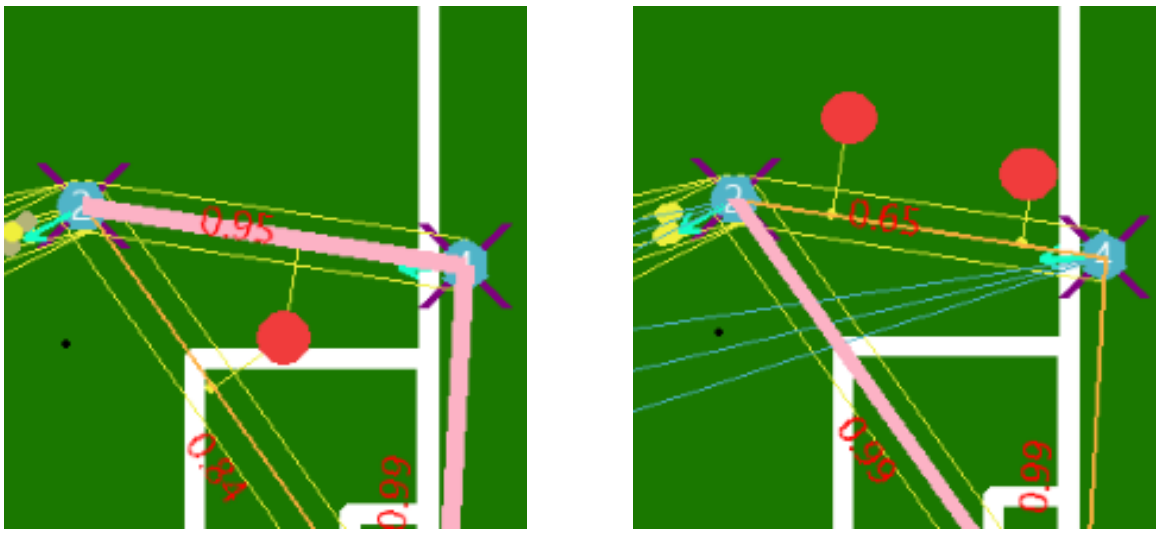
Variable Name	Meaning
$dist_{bR}$	Distance between robots
x_{ri}	Coordinate X from robot number i
y_{ri}	Coordinate Y from robot number i
p	Constant factor to normalise the Bayesian curve between $[0, 1]$
d	Bayesian curve deviation - margin used for the ideal action distance
$BDTA$	Best distance to action (pass or goal), ideal action distance between two robots
tp	Total probability - action given probability
$prob$	Total probability 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
S_{robot}	Opponent's robot size

$$dist_{bR} = \sqrt{(x_{r1} - x_{r2})^2 + (y_{r1} - y_{r2})^2} \tag{7.1}$$

$$tp = \frac{p}{d\sqrt{2\pi}} e^{\frac{-(dist_{bR} - BDTA)^2}{2d^2}} \tag{7.2}$$

Equation (7.2) is based on a normal distribution curve [20, 37]. 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 (7.3). This influence is only calculated if the opponent robot is between the robots being analysed and with a distance to the line of pass smaller than $dist_{start\_infl}$. It is also influential the size of the robot S_{robot}, as this also affects how much interference the opponent robot has on a line of pass. Figure 41 (a) shows an opponent affecting two lines, one of pass and one of goal. Figure 41 (b) shows an example of a line of pass being influenced by two opponents.

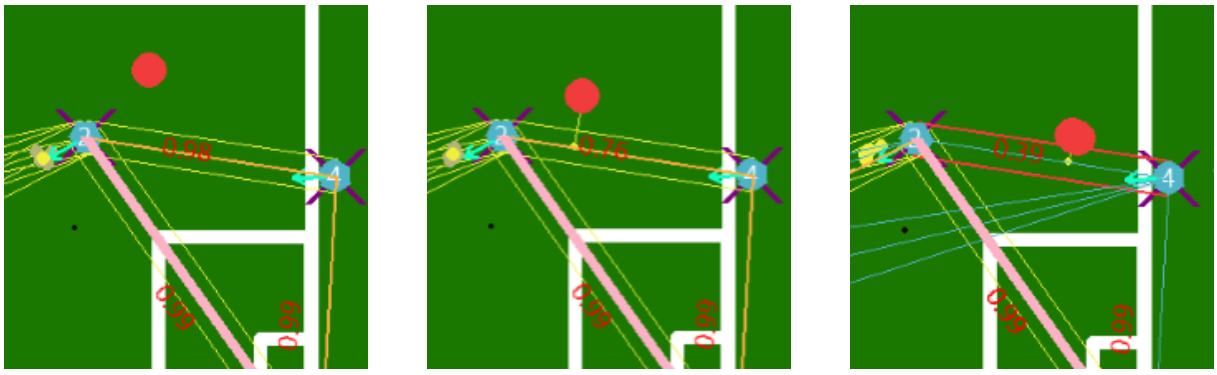
$$prob = tp * \frac{dist_{Opp\_to\_Line}}{S_{robot} * dist_{start\_infl}} \tag{7.3}$$



(a) (b)

Figure 41: Opponent's influence: (a) One opponent affecting two lines; (b) Two opponents affecting one line of pass

Figure 42 shows a possible pass with a high success probability and the influence of an opponent getting near the line of pass. In Figure 42 (c), the polygon surrounding the line of pass is the limit of a possible pass. If the opponent intersects that boundary, the pass is not considered an available line.

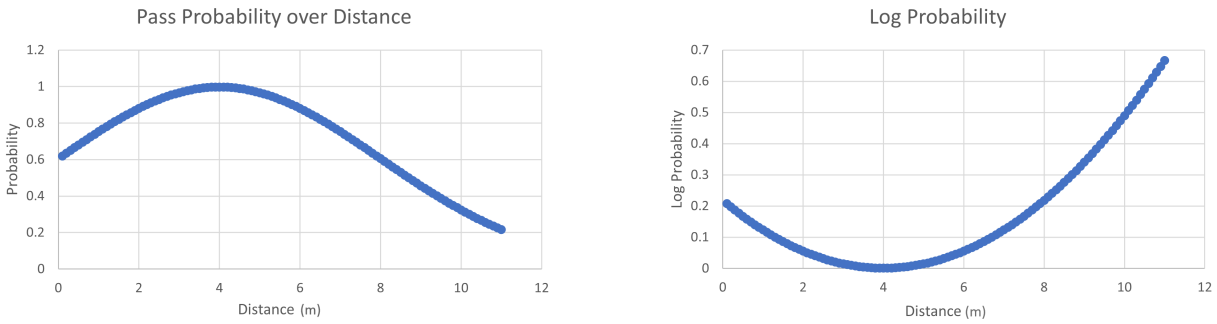


(a) (b) (c)

Figure 42: Pass probability with the opponent's influence based on its distance

7.5.2 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 [36] 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 (7.4), with values constrained between 0 and 100. This upper limit serves to prevent computational errors and ensures the desired precision for operational efficacy. Figure 43 shows the Bayesian curve of an ideal pass probability over distance (a) and the respective log probability (b).



(a) (b)

Figure 43: Log probability conversion: (a) Pass probability over distance; (b) Log probability of pass over distance

$$v = -log_{10}(prob) \tag{7.4}$$

This theorem is used for multiple reasons. For computation, addition is much simpler and faster than multiplication. In this case, to study different paths, all the calculations use probabilities. The probability of $A \wedge B$ would be $P(A) * P(B)$, but with the logarithms the product becomes a sum [36]. 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 $]-\infty, 0]$ [36] and therefore the floating point resolution limitations do not exist.

7.6 Graph Generation

Regarding the graph that is generated based on the probabilities calculated, this is a complete graph [19, 48], with n_{nodes} nodes, and the number of connections (n_{con}) is given by Equation (7.5) [41]. 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 score. 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.

$$n_{con} = \frac{n_{nodes} * (n_{nodes} - 1)}{2} = \frac{(n_r - 1) * n_r}{2} \tag{7.5}$$

An example of a complete graph in a game situation is visible in Figure 44. 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 visualisation and interpretation purposes.

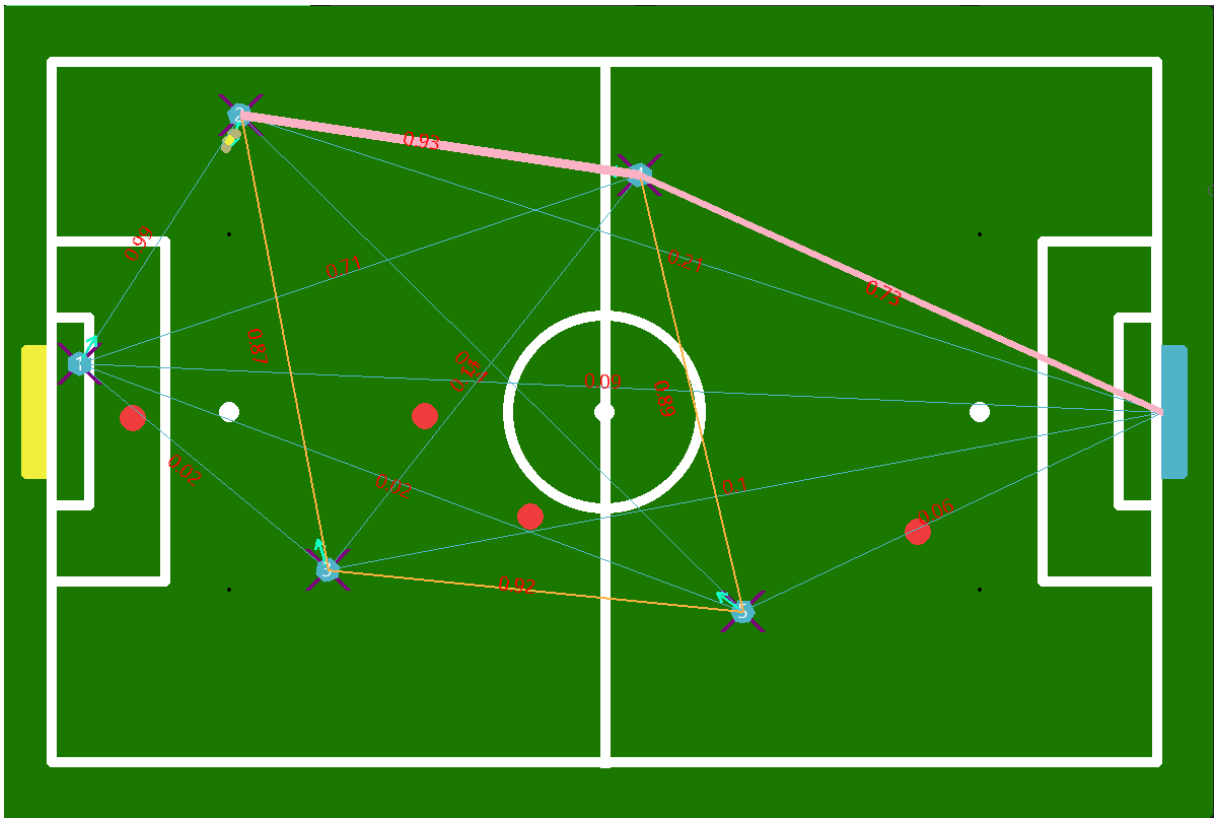


Figure 44: Complete graph in a game situation

7.7 Ideal Probability Path Finder Algorithm

The calculation of the highest probability path (lowest log probability value) is based on the path finder algorithm A^* [49, 42]. The input to the algorithm is the graph itself with the starting and ending nodes. These nodes are always the robot that has the ball (starting node) and the goal (ending node). 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. The algorithm runs at every planning iteration and outputs the set of actions that maximises the probability of scoring a goal. When the overall probability is low, the kicking skill is delayed in order to give the team more opportunities to find a better set of actions, which occurs thanks to the positioning of the robots. The highest probability path is represented in pink lines in the graph, 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 in situations when the game changes very

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.

7.8 Decision Tree

The Play Generator covers the desirable path 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 [39, 40], and for this strategy, there are two of them.

7.8.1 Game Decision Tree

The game decision tree handles the game state and decides the skills desired for the current game situation, as shown in Figure 45. 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 in section 7.8.2 Player Decision Tree). The tree can be readjusted and configured by a JavaScript Object Notation (JSON) config 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 6.

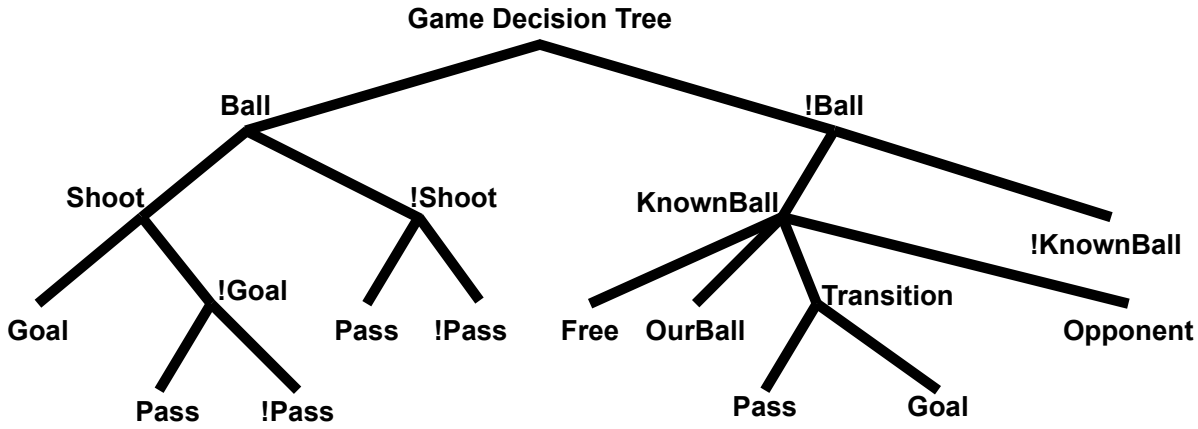


Figure 45: Game-state decision tree

Table 6: 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 between actions, moving after a kick
Opponent	Opponent team has the ball or is closer
Pass	The transition is a pass from own team
Goal	The transition is a goal kick

7.8.2 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 46. It is based on their current strategy, game and time states.

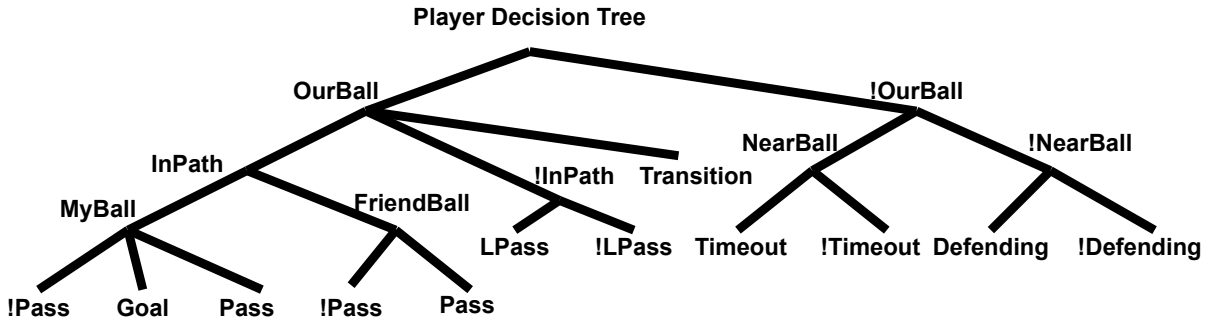


Figure 46: Player decision tree

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 7.

Table 7: 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

7.9 Robot Positioning and Heat Maps

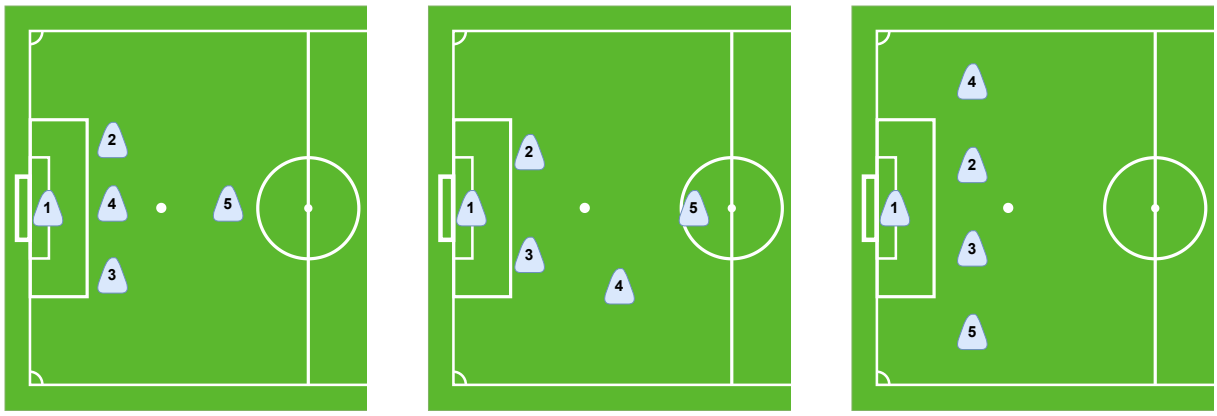
Equally important to the decision-making, is the robot's positioning. Being in the right location at the right time has a huge impact on the game. Following the same logic as the decision-making, and with the need to have different outcomes for different situations, the use of weights and probabilities was the approach taken.

7.9.1 Zones

A baseline positioning setup is important to always have safe and known options, and therefore the same solution applied in real football is used here. Formations serve as the blueprint for tactical organisation on the field, influencing a team's performance and strategy. They organise the player's positioning, movement, and overall team structure during the game. Formations in futsal optimise both defensive and attacking situations, team coordination, and fluid transitions between offence and defence. Effective formations require synchronisation among players, and based on that, the playing style can significantly impact match outcomes.

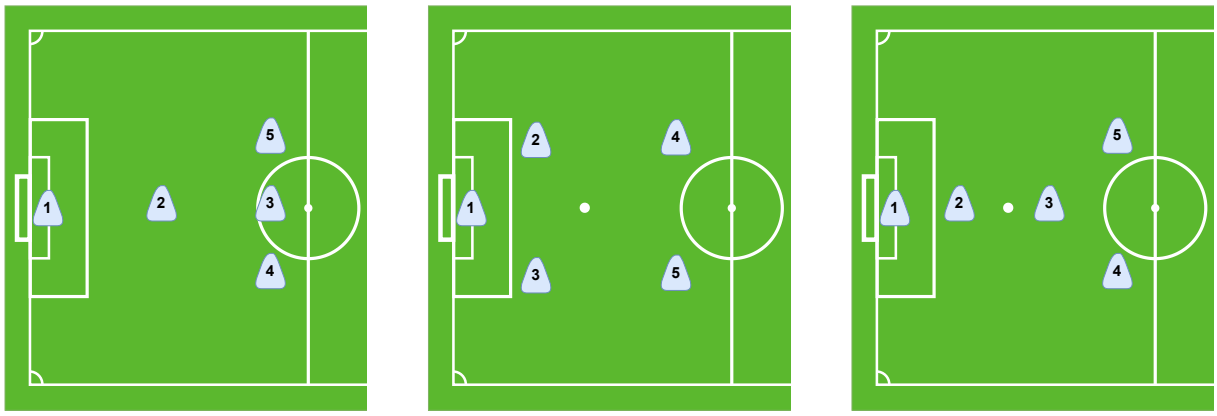
Futsal Zones

Futsal has multiple known formations and is a 5v5 game just like MSL. Multiple possible futsal formations were studied, and the most common are presented in Figure 47 and 48. All of them have advantages and disadvantages and all of them are used in different games against different opponents.



(a) (b) (c)

Figure 47: Futsal formations: (a) Wall; (b) Side Line; (c) Line



(a) (b) (c)

Figure 48: Futsal formations: (a) All or Nothing; (b) Square; (c) 'Y' formation

Zones in Robots

When applying the formations to the robots, one of the most effective formations when balancing both offensive and defensive situations is the Diamond formation. This formation has two configurations possible for defensive and offensive situations, visible in Figure 49.

This formation is based on a series of symmetric triangles that translates to multiple lines of pass simultaneously and provides effective cover when defending a certain player. The biggest disadvantage of this formation is the fact the players are required to move substantially. Which for a robot, contrary to a human, only stops when runs out of battery.

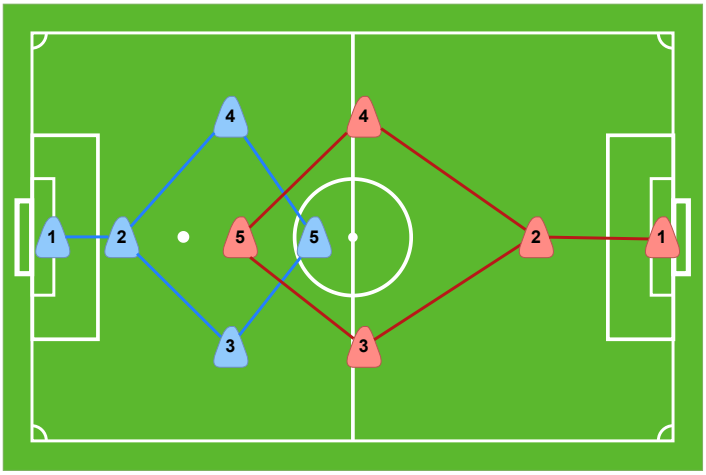


Figure 49: Diamond formation offensive and defensive situations

An important difference between futsal and MSL is that robots know and can do all the same things. Robots are built equally, and all can move and control the ball the same way. That is a difference from the human game that can be taken advantage of. When using a Diamond formation, humans run a lot to keep every opponent under pressure and to keep their position at the same time. Since all the robots are equally built, it is possible to take advantage of that by not having fixed positions in the formation assigned to each robot. After a play, robots do not need to return to their assigned position or formation due, because another robot can replace them and swap positions.

Zone Assignment and the Hungarian Algorithm

The Zone assignment is solved by exploiting the indifference between robots. The simplest yet most effective solution is assigning zones based on distance. Every robot should be assigned to its nearest zone, but two robots cannot be assigned to the same zone and no zone can remain without a robot. Zone assignment is based on the shortest distance overall, so the sum of the distance between every robot and its assigned zone needs to be as low as possible. The Hungarian Algorithm solves that problem.

The Hungarian algorithm is used to solve an assignment problem with the optimal combination of agents to tasks while minimising the overall cost, in this case, distance. A matrix is created with the distances between every robot and zone. It iteratively identifies the most efficient assignment by seeking a combination of assignments that collectively have the lowest distance.

Figure 50 shows two cases that are very similar but have different solutions because the overall distance is different. Robot number one, the goalkeeper, is not taken into consideration in zone assignment because it is not desirable for it to leave the goal.

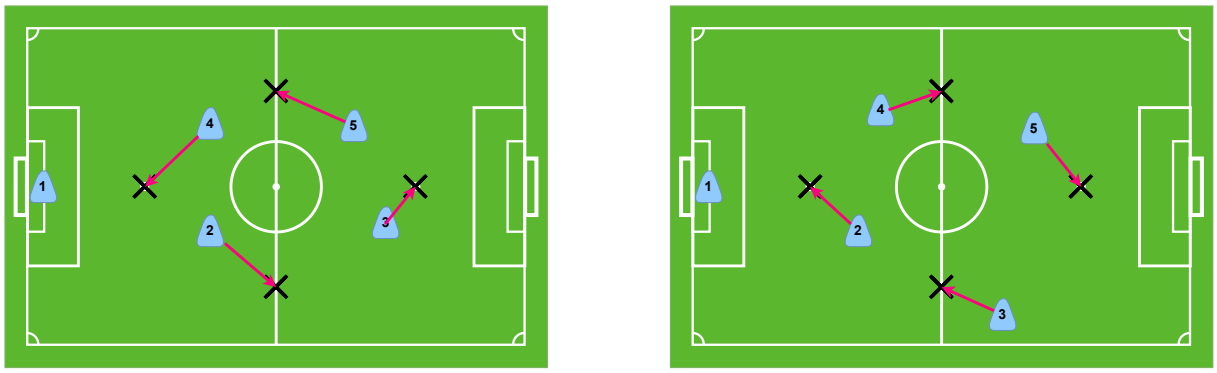
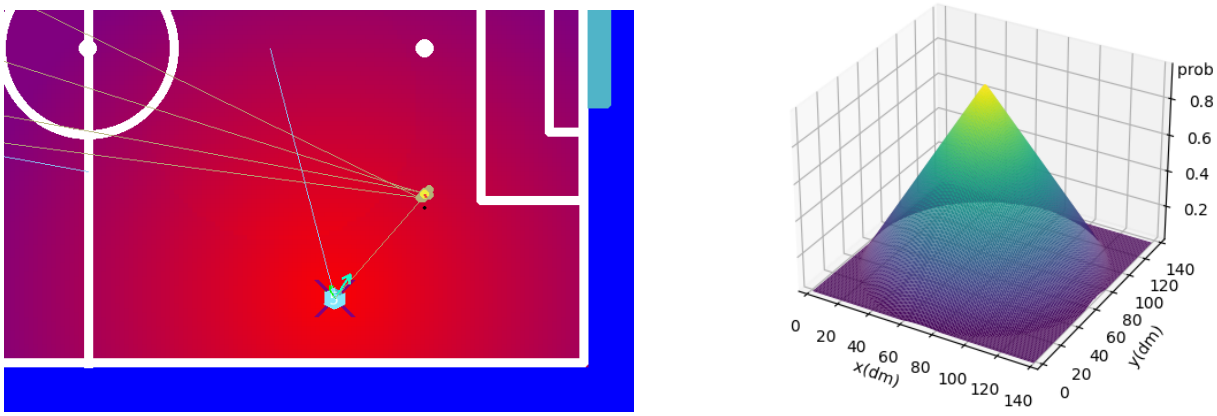


Figure 50: Hungarian algorithm zone assignment example

7.9.2 Base Maps

The sum of multiple base maps that are dynamically adapted for every game iteration is each robot heat map. Robots positioning is based on each robot heat map. The positioning map is calculated for each robot, which is the fusion of multiple distinct base maps, each with pre-generated matrices of weights. These matrices represent specific game influences and are created using distinct formulae aligned with the desired outcome. The heat map colour scheme is a "Red-Blue" format, where the blues represent low-value areas and the red represents high-value areas. It can be associated with warm and cold zones, where the robot always prefers the warm zones (red) and avoids the cold zones (blue). The representation formulae and user interface is presented in 8.1.3 Heat Maps from the Chapter Base Station.

Base maps can be either entire field or partial field maps. They can be team or single robot maps, and they can either be influence maps or a rule. Entire field maps have influences that, based on a static matrix, impact the whole field regarding the robot's position. Partial field maps are mostly robot-focused and readjusted at every game iteration. Team maps are calculated for everyone on the team. For example, the influence of all opponents is the same regarding the robot's position. Everyone on the team can use that map after being calculated since it does not have the robot's position as an input. Single robot maps are calculated for every robot, as is the example of the Distance Map presented in Figure 51. This map is calculated based on the distance between the robot's current position and its surrounding locations. This particular map serves the purpose of limiting the movement options available to players by excluding long-distance solutions. In fact, this distance base map ensures a more focused range of movement for the player, eliminating choices of longer movements or time-consuming.



(a) (b)

Figure 51: Representation of a base map based on the distance to the robot: (a) 2D base station representation; (b) 3D representation

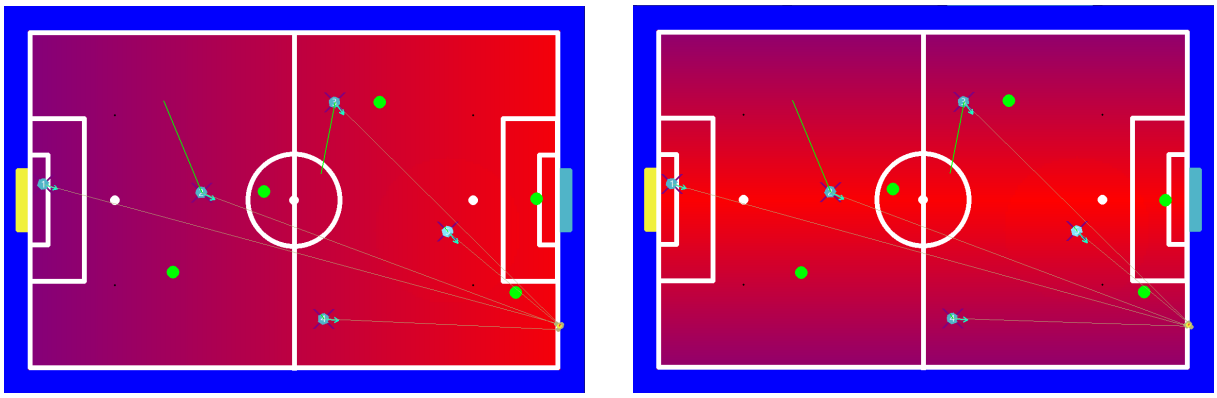
This example is a linear equation based on the distance, visible in Equation 7.6 where C_x and C_y are the matrix centre coordinates, x and y are all the points from the matrix. β in Equation 7.7 is a multiplication factor used to normalise the values between matrices. The radius of the distance influenced is defined by r. The base map is achieved by calculating α for every matrix position.

$$d = \sqrt{(x - C_x)^2 + (y - C_y)^2} \tag{7.6}$$

$$\alpha_{x,y} = \begin{cases} 1 - (\frac{\beta * d}{r}), & \text{if } d \leq r \\ 0, & \text{otherwise} \end{cases} \tag{7.7}$$

Forward Field and Centre Field

Figure 52 shows two base maps with similar purposes but different influences. These are entire field maps, and the first (a) influences to make the robots move forward in the field. It makes all the solutions have a forward and progressive approach to the field. The second map (b) is to make the robots avoid the sidelines. It has the purpose of making the robots play more in the middle of the field, to avoid losing the ball on the sideline. This map is important, especially for a new team with control systems that are not fine-tuned.

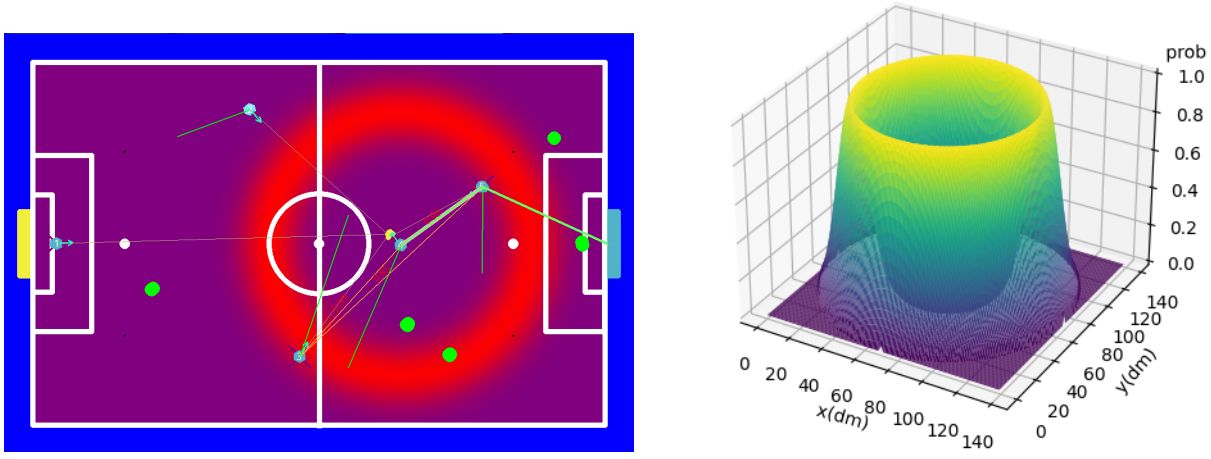


(a) (b)

Figure 52: Representation of base maps: (a) Forward field; (b) Centre field

Ideal Pass Distance

One way to optimise the pass probability is to position the robots in ideal conditions of 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 53, (a) is a field representation, and (b) is the Bayesian curve in a 3D representation [50]. For the robot in possession of the ball, the centre 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 centre 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 probability in Equation 7.2.



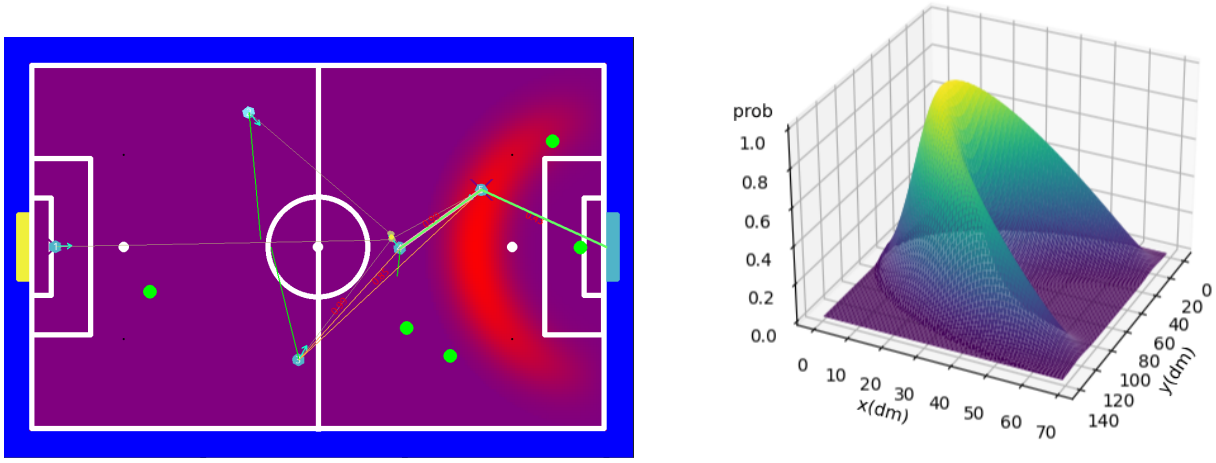
(a) (b)

Figure 53: Ideal pass distance base map: (a) Field representation; (b) 3D pass probability representation

Ideal Goal Distance

Just like in the ideal pass distance base map, the same principle is applied to the ideal goal distance, but with some changes because of the importance of the kick angle between the robot and the goal. When performing a pass, both robots are facing each other. When kicking to the goal, the angle from where one shoots influences the area visible on the goal. Equation 7.8 is the formula for the ideal goal distance, which is the pass probability Bayesian curve times the cosine of the angle α between the goal and the kicking robot. 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. Contrary to other base maps, this matrix is stationary and is always applied to the centre of the opponent's goal. Visible in Figure 54 is the matrix applied to the field (a) and the 3D representation of the matrix (b).

$$GoalProb = cos(\alpha) * \frac{p}{d\sqrt{2\pi}} e^{\frac{-(dist_{RG}-IGD)^2}{2d^2}} \tag{7.8}$$



(a) (b)

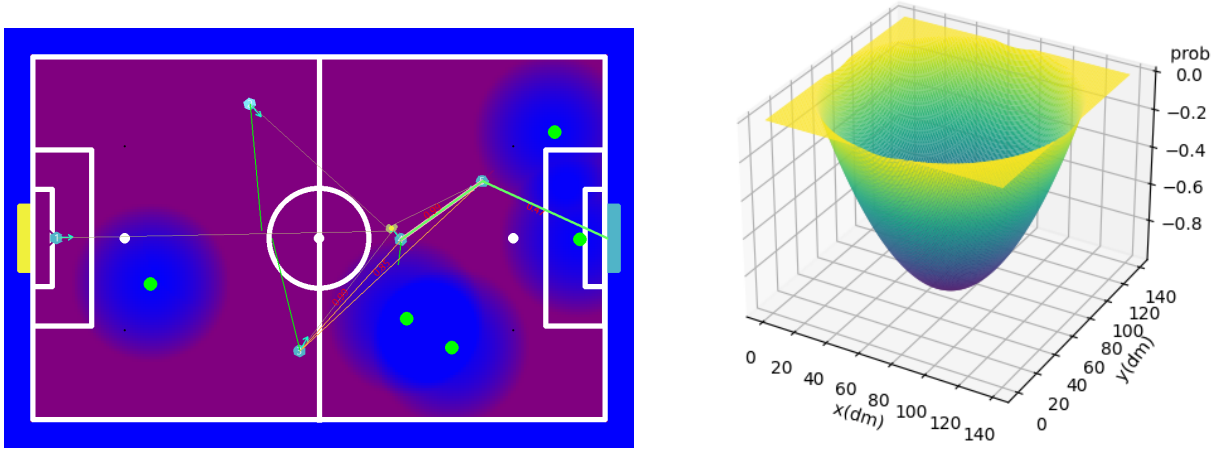
Figure 54: Ideal goal distance base map: (a) Field representation; (b) 3D goal probability representation

Opponents Circle

In an offensive situation it is not ideal to be near the opponents, and that is the reason for this base map. It consists of a simple negative power function to make the robots avoid the opponents. The matrix is the same for every opponent. For optimisation reasons, and because the opponent's influence is the same for every robot, one map is created with all the opponent's matrices. The full map is the combination of n matrices, being n the number of opponents detected. Figure 55 (a) represents the field with all the opponent's matrices, and (b), is a 3D representation of an individual matrix with its formula visible in

Equation 7.9. Just like on other matrices with circular shapes, all the values outside the radius defined as r are cleared to 0.

$$\alpha_{x,y} = (\frac{\beta * \sqrt{(x - C_x)^2 + (y - C_y)^2}}{r})^2 \tag{7.9}$$

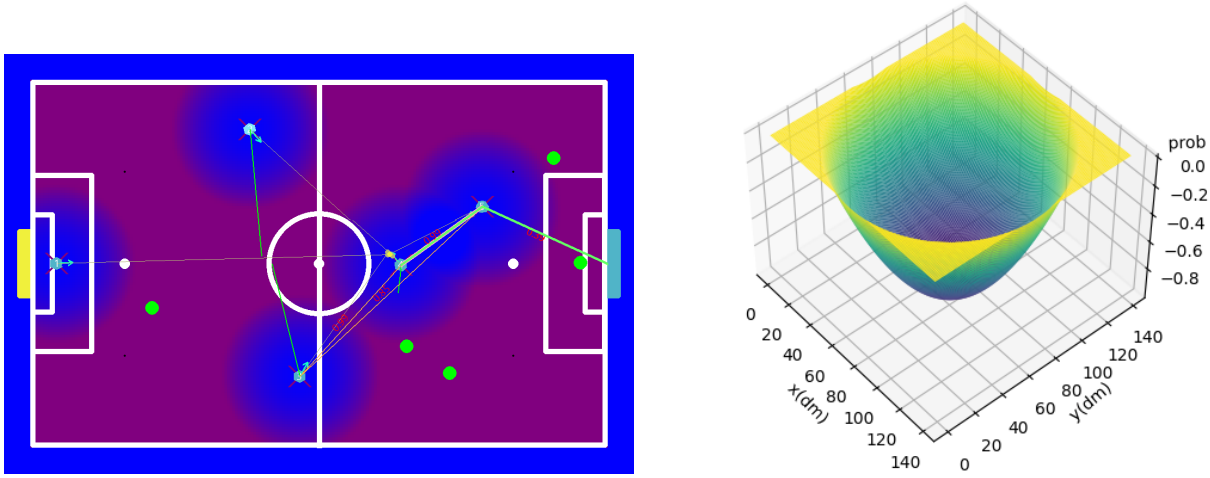


(a) (b)

Figure 55: Opponents influence: (a) Field situation; (b) Individual matrix 3D graphical representation

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 56 is a field map (a) and the matrix used for every teammate (b). The formula used is the same as the Opponents Circle Map, as shown in Equation 7.9, but with different β and r values.



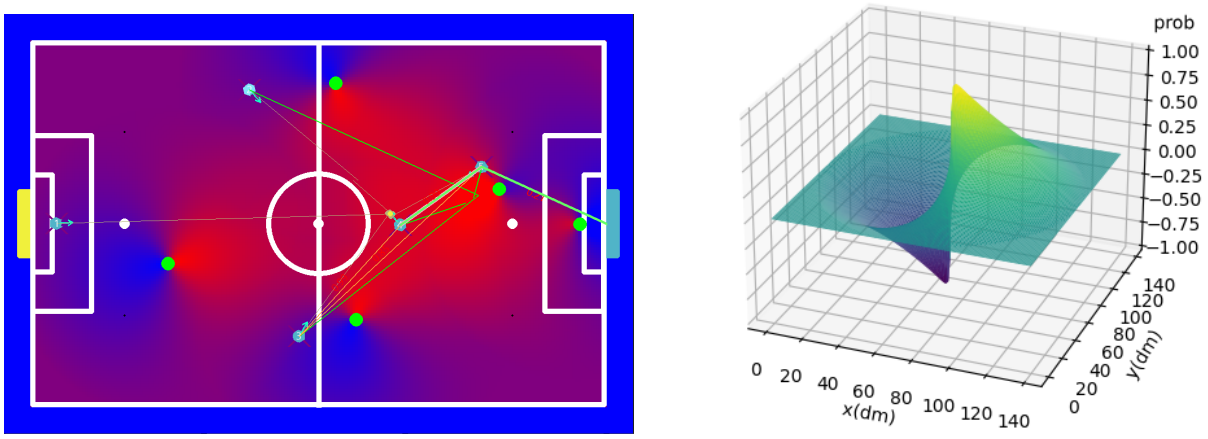
(a) (b)

Figure 56: Teammates influence: (a) Field situation; (b) Individual matrix 3D graphical representation

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 57 (b) [51]. For optimisation reasons, and because there are 360 possible matrices (due to the dependence on the angle between the opponent and the ball), these matrices are all previously calculated and then just used to create the field in every situation. For the same reason as the teammates and opponents showed influence, this map is generated with all the opponents present, and used for all teammates, visible in Figure 57 (a). The formula for the matrix used is a cosine of the angle between the opponent and the ball times the distance to the opponent itself. In Equation 7.10, ϵ is the angle between the opponent and the surrounding points, C_x and C_y are the opponent coordinates, x and y are the relative coordinates around it and r is the maximum radius affected by the field.

$$\alpha_{x,y} = cos(\epsilon) * (\sqrt{(x - C_x)^2 + (y - C_y)^2} - r) \tag{7.10}$$

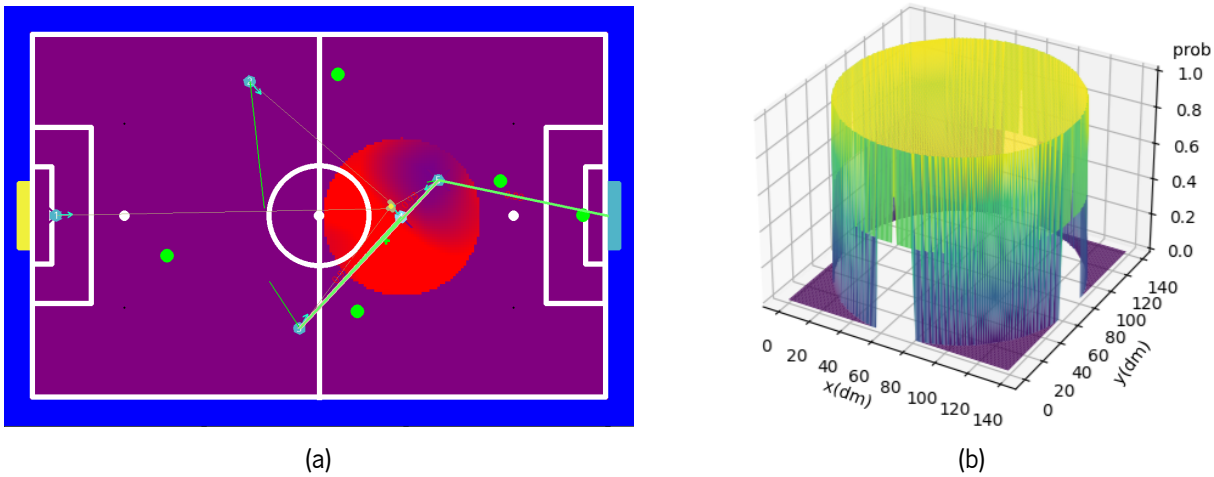


(a) (b)

Figure 57: Representation of a base map based on the distance to the robot: (a) 2D representation; (b) 3D base station representation

3-Meter Radius with the Ball

To make the games more appealing and to increase the robot's cooperation, there is a rule that limits the amount of space a robot can move with the ball (3-meter radius). The rule can be applied to the strategy using a map that limits how much a robot can move when in possession of the ball. The matrix of this representation is presented in Figure 58 (b) where all the values inside the 3-meter radius are 1 (one) and all the other ones are 0 (zero). This rule map is applied as soon as a robot gets control of the ball. The centre location of the matrix is set in the moment of the robot grabbing the ball, and stays still until the robot no longer has the ball. 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. After summing all the fields, this field is multiplied by the others to limit the robots' movement. A game situation of a player who has the ball is available in Figure 58 (a).



(a) (b)

Figure 58: 3-meter radius rule map: (a) Field situation representation; (b) 3D matrix representation

Outside Field

Another rule base map is the outside field. All values outside the field are set to the lowest value possible to make the robots avoid it. A rule like the 3-meter radius map is applied by multiplying it to the sum of all other influence maps. This map is visible in all the previously shown game situations.

7.9.3 Calculation of the Robot Map

As mentioned before, some maps are entire field maps, while others are location-specific. They all need to be translated so that a sum of the maps can be made. Partial field maps are translated to match the same matrix size as the field to optimise its summing process. That is achieved by placing the desired matrix on the robot's location. In this case, a python library for matrix manipulation is used called *numpy* [52], and this transformation process is achieved by the `numpy.pad` function that does exactly what is visible in Figure 59.

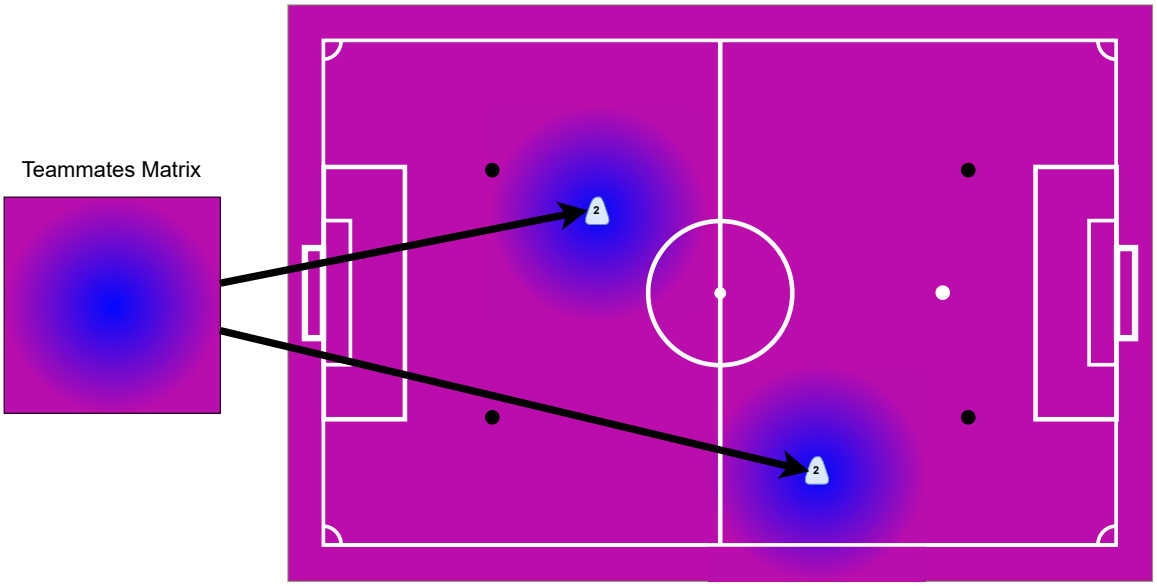


Figure 59: Matrix size adjustment for non-entire field maps

Each map has associated an ω value that is adjusted for different game situations. The weight of the matrix varies from game and player situation defined by the Decision Trees (7.8 Decision Tree). This weight ω is multiplied to every value in the matrix. With all the matrices the same size, its fusion can be achieved by summing all the maps. Each robot's influence heat map M is achieved by summing all maps, where each map is multiplied by its weight's influence ω, visible in Equation 7.11.

$$M = \sum_{i=0}^{n-1} A_i \omega_i = \begin{bmatrix} \sum_{i=0}^{n-1} a_i(0,0)\omega_i & \sum_{i=0}^{n-1} a_i(1,0)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,0)\omega_i \\ \sum_{i=0}^{n-1} a_i(0,1)\omega_i & \sum_{i=0}^{n-1} a_i(1,1)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,1)\omega_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{n-1} a_i(0,y)\omega_i & \sum_{i=0}^{n-1} a_i(1,y)\omega_i & \cdots & \sum_{i=0}^{n-1} a_i(x,y)\omega_i \end{bmatrix} \tag{7.11}$$

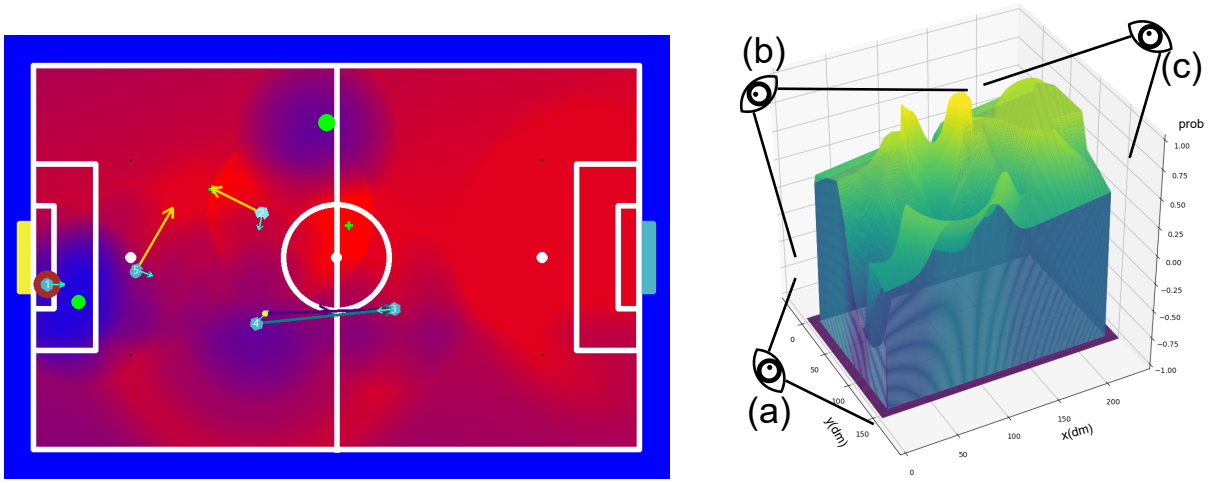
The variable $a_i(x, y)$ represents the multiple base map matrices, $A_1, \ldots, A_i$ respectively, and ω_i is the associated weight of each map influence.

After the fusion of all the influence maps, the rule maps still need to be applied. These maps are also transformed into the right size and then applied to the final map by multiplying the values of every position in the matrix. Fm is the final map with all the influences and rules, visible in Equation 7.12 where $R_1, \ldots, R_i$, are the rule base maps and $R_0 = M$ previously calculated in 7.11.

$$Fm = \prod_{i=0}^{n-1} R_i \omega_i \begin{bmatrix} \prod_{i=0}^{n-1} r_i(0,0)\omega_i & \prod_{i=0}^{n-1} r_i(1,0)\omega_i & \cdots & \prod_{i=0}^{n-1} r_i(x,0)\omega_i \\ \prod_{i=0}^{n-1} r_i(0,1)\omega_i & \prod_{i=0}^{n-1} r_i(1,1)\omega_i & \cdots & \prod_{i=0}^{n-1} r_i(x,1)\omega_i \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{i=0}^{n-1} r_i(0,y)\omega_i & \prod_{i=0}^{n-1} z_i(1,y)\omega_i & \cdots & \prod_{i=0}^{n-1} r_i(x,y)\omega_i \end{bmatrix} \tag{7.12}$$

7.9.4 3D Robot Positioning Map

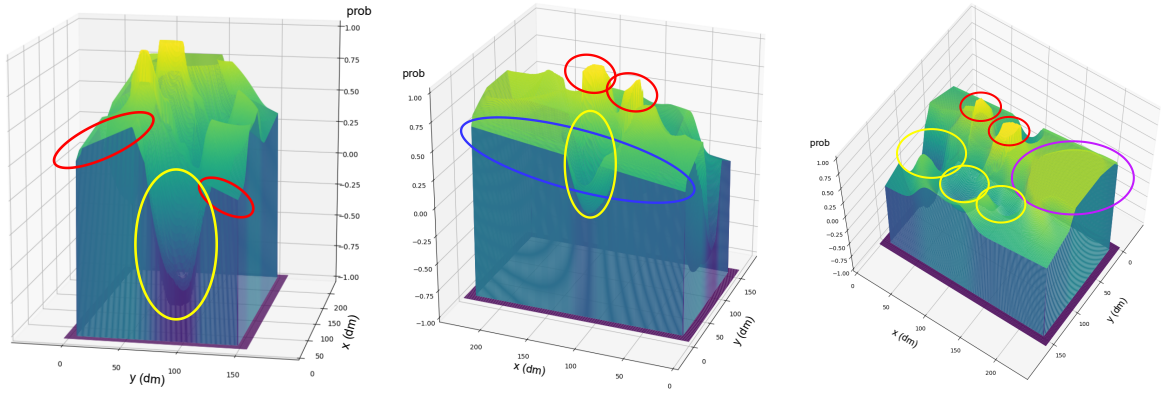
In order to better understand the influence of each separate base map, a 3D representation of one heat map was created. Figure 60 shows the map in its 3D normal representation (a) and its 3D representation (b) where multiple influences can be observed. Figure 60 (b) has three points of view, which are visible in Figure 61.



(a) (b)

Figure 60: 3D robot positioning map: (a) 2D representation; (b) 3D representation

Figure 61 (a) shows 2 different influences. The centre field, which has the influence of making robots play and avoid the sidelines, is marked in red. Annotated in yellow is an Opponent Circle map in order to avoid being near them since the team is in an offensive situation. With the same yellow mark, Figure 61 shows another opponent's influence that is present at the top of the field near the sideline. Noted in red are the maximum probability points, which are marked in Figure 60 (a) with small green crosses. With a blue ellipse, it is shown the Forward influence that makes robots progress on the field. Figure 61 (c) shows in yellow the teammate maps, which have low values in order to make the team spread in the field. In red are presented again the two maximum probability points, and in purple, are shown the ideal goal influence.



(a) (b) (c)

Figure 61: 3D points of view: (a) Goal view; (b) Side line view; (c) Top view

7.9.5 Position Calculation

A heat map does not return a specific position but a general area of where the robot should go. 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 (one) and the rest is 0 (zero). This binary map is then used to detect all the good positioning areas available. To calculate the position based on the heat map, it was used the *OpenCV* library [53]. With the functions `cv2.findContours` and `cv2.moments` the centre of mass is calculated for every area.

When multiple ideal areas are available, the selection of which one to choose is based on the largest area. Figure 62 shows a situation of four maximum value areas, with their specific maximum value centre of mass with a green cross, and its binary map. After calculating the centre of mass of all four, the location chosen is the one with the highest area, in this case, location B.

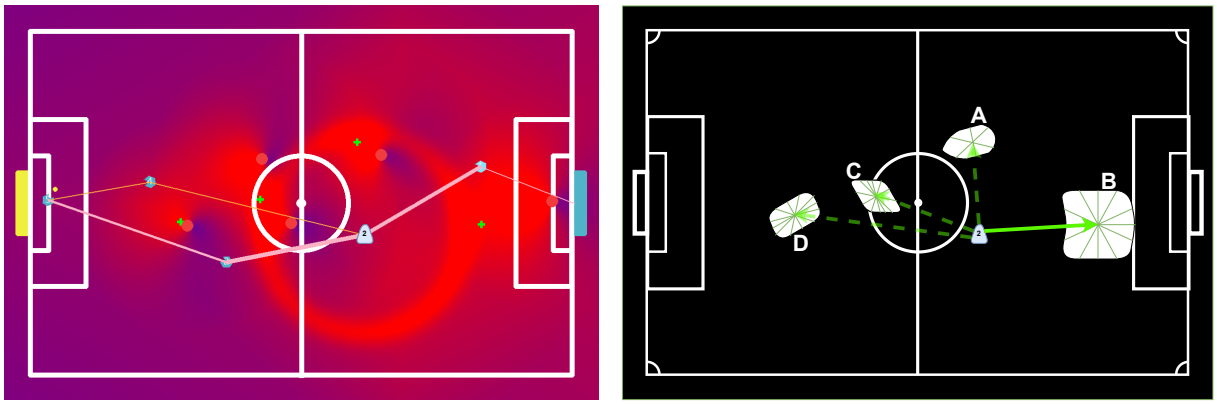


Figure 62: Position selection based on multiple possible areas

7.10 Skill Assignment

The outcome of the Player Decision Tree and Game Decision Tree gives a group of five skills to be assigned to the team robots. These assignments are based on a couple of factors that vary between distance to the goal, current position, distance to the ball and even pressure variables.

7.10.1 Offensive Situation

In offensive situations, almost all assignments are solved based on the ideal probability path. The robot who has the ball should pass to the next robot in the path and so on, until one robot kicks to the goal. All the robots not present in the ideal path should move to do one of two things: stay in the defensive

area and find a position to be part of the probability path; giving alternative lines of pass. In every situation, the goalkeeper must remain in the defend skill to protect the goal.

7.10.2 Defensive Situation

Regarding defensive situations, the same approach as futsal is used. There is always at least one robot responsible for putting pressure on the opponent who has the ball. The other three strikers have an assigned opponent to defend with the Cover skill. The robot that should have the Attack skill to put pressure on the opponent with the ball is the one nearest to the ball. The opponents are assigned to each robot using the Hungarian algorithm. This algorithm is applied with the robots that are not in the attack nor the defend skill. This is verified in every iteration. If opponents swap positions, the robots also swap their assigned opponent and remain in the Cover skill to block them from receiving the ball.

7.10.3 Skill Arguments Calculations

For each skill, arguments are calculated differently depending on the situation. The arguments in the Defend, Receive and Attack skills are the ball location, and it is given to the robot the weighted average calculated by the data fusion. Robots will only use the sent ball location in case they do not detect it with their own vision system. In the kick skill the arguments of where to kick are always the location of the next node in the ideal probability path, that can be either a teammate or the goal. The P/K argument is set depending if it is a Pass or a Kick to the goal. The additional displacement arguments D_x and D_y are calculated based on the ideal position of the skill calculated by the heat maps. In the Cover skill, the arguments are the two objects one should cover. This is only used in defensive situations, and the points are the assigned opponent to defend, given by the Hungarian algorithm used and the goal. Regarding the Move skill arguments, they are calculated by the heat maps. Its orientation is set to be in line with the ball to take advantage of the front-facing camera.

7.11 RefBox Situations

RefBox is the referee's computer, that is used to send the referee decisions onto each team's base station. All robots must obey the RefBox commands, and in case some robot malfunctions and does not behave properly, it will be considered a damaged robot and must leave the field. All referee interactions start with the Stop button from the RefBox. Therefore, when that button is pressed, the skill Stop is sent to all robots. After the ball is placed in the desired position by the referee, one of the situation buttons is

pressed. The possible situations are kickoff, free kick, goal kick, throw-in, corner and penalty, which can be to either team.

In a RefBox command for the opponent team, robots will behave as they normally would by the strategy, with one key difference. The player decision tree that selects the skill for each robot will not allow any player to use the Attack skill before the referee restarts the game. The robots will just position themselves trying to cut lines of pass and put pressure on the opponent team.

Regarding a RefBox command for an offensive situation, where the team has the ball possession, robots will also behave as the strategy dictates in an offensive situation. The player next to the ball that will restart the game, will not receive the Attack skill before the referee restarts the game. Also, in the first Attack skill to restart the game, the striking robot will receive a flag in the arguments that dictates the robot to move slowly to the ball in order to not take it out of place.

To summarise, in RefBox commands robots will still detect the ball and opponents. When the referee instructs the robots which offensive or defensive commands it is, the robots will move accordingly to effectively restart the game.

Chapter 8

Base Station

The base station computer has the responsibility for numerous modules, each with distinct tasks relevant to debugging, gameplay, calibration and testing. The development speed was a crucial factor, and consequently, Python was selected as the programming language. Besides its fast development time, Python has excellent graphical and data representation tools.

Pygame was the library used for Graphical User Interface (GUI) development due to its graphics/multimedia features, performance, active community and open-source nature. Because of its game-oriented focus, the use of the Simple DirectMedia Layer (SDL) allows the pygame to take advantage of hardware acceleration for fast graphics rendering.

8.1 Tabs and their Purpose

Using the same GUI as a browser, the base station interface is divided into tabs that serve different purposes. With a total of five tabs, these are the easiest ways to organise the representation and show different GUI elements. The tab division also allows the computer to render only the information needed for that specific tab.

8.1.1 Game + RefBox

The primary interface is the "Game + RefBox" tab which plays a central role in this system. It is used during the game and displays both the ongoing match and vital robot information. It shows the field, the robots and the detected elements' representation. Below the field, there are five boxes with each robot's information updated in real-time (around 25 times a second). This is where all the non-game robot information is presented. The exact information received is presented in Table 3, section 5.3 Data Frames from Communication System chapter and more information about how the data is presented in section 8.2.2 Robot Information.

There are buttons next to the field that simulate the RefBox for testing purposes without the need for a separate Refbox computer. The current score is prominently displayed in the top-right corner. For a clear overview, this interface's visual representation is shown in Figure 63.

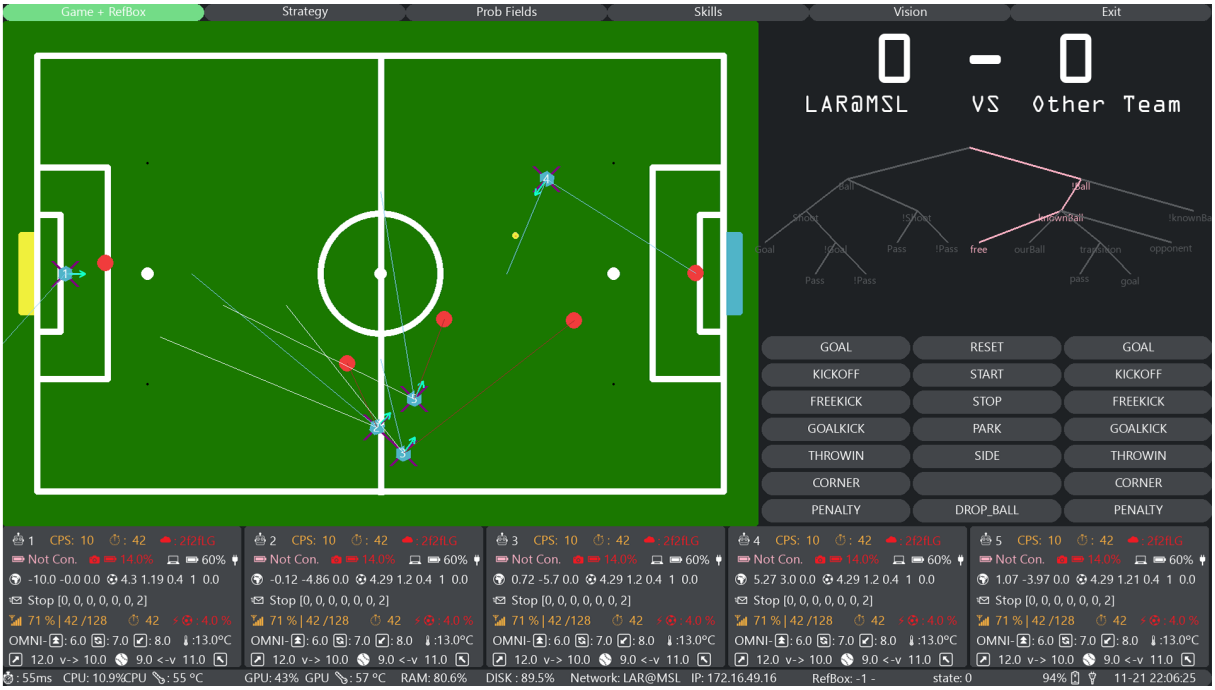


Figure 63: Game + RefBox tab

8.1.2 Strategy

The "Strategy" tab serves as a debugging tool for the decision trees within the game strategy, visible in Figure 64. Similar to other tabs, it displays the robot information on the bottom. This tab includes both the Game Decision Tree and the Player Decision Tree on the right. The active state within the decision trees is in pink. The representation of the decision trees is modular and based on the configuration files imported at the beginning of the base station code.

Users have the flexibility to choose which player's decision tree to display by using the keyboard numbers one to five to select the desired robot. The robot selected has its information box with a different colour in order to show to the user the selected robot. In Figure 64 the selected robot is number one, so the player decision tree shown is from robot number one.

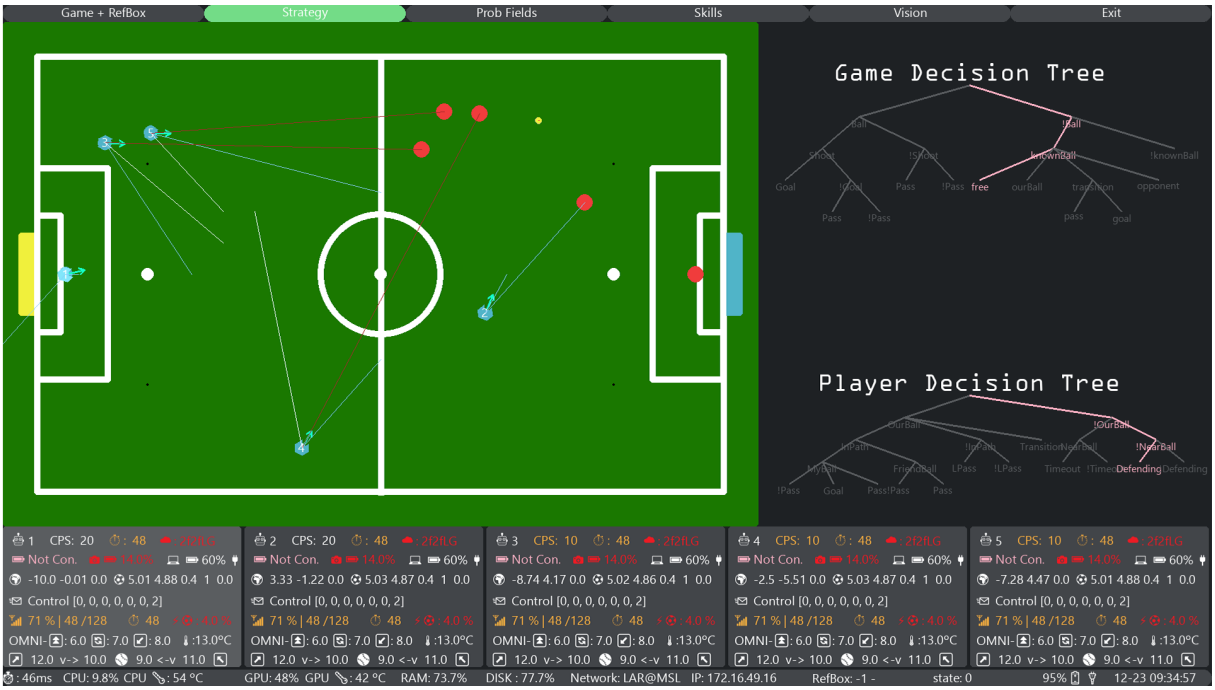


Figure 64: Strategy calibration tab

8.1.3 Heat Maps

The "Heat Maps" tab, visible in Figure 65, is essential for debugging, calibration, and visualisation of positioning maps. Instead of the conventional green field, it represents the probability maps used for positioning with a colour-coded matrix. The representation shows the complete map of the selected player, number five in this case, decision tree state, with high-probability areas marked in red and low-probability areas in blue. To calibrate a map other than the current player's state, users can enable the checkbox on the right and select the desired map from the dropdown menu. Each dropdown menu option is a possible outcome of the player decision tree discussed in 7.8 Decision Tree in Game Strategy chapter. Users can adjust the weight of each base map using the track bars on the right side to calibrate the influence based on the current player's situation.

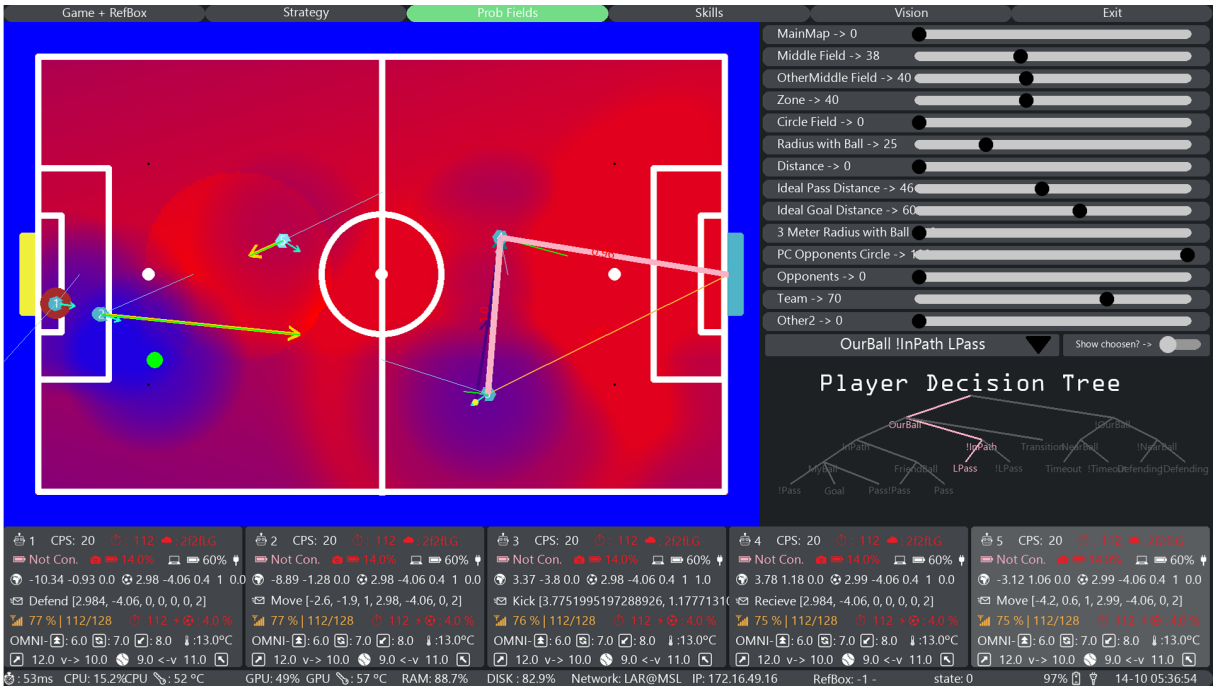


Figure 65: Heat maps tab with calibration trackbars and player decision tree

8.1.4 Skills

The "Skills" tab plays a crucial role in calibrating control systems as well as any team demonstration. In this tab, users can select a specific robot using the keyboard numbers one to five, place markers on the field, and assign a specific skill for the chosen robot to execute. With this interface, one can control and test certain skills without the need to reset or manually move the robots.

The markers are used to determine the skill arguments. For example, the kick skill needs an argument specifying the target location, and in this tab, such arguments are calculated using the markers. On the field, the markers are represented as pink and brown circles, such as the first and second markers, respectively. The need for two markers is due to certain skills requiring more than one location argument to be executed. Figure 66 shows various skills that were manually sent to the robots using the markers and the skill buttons. Robot one is in the Stop skill whereas robots number two and four are in the Move skill. Robot number three, the selected one, is in the Cover skill. The markers for the selected robot, number three in this case, are also visible.

On the right side, below the skills buttons, a set of track bars is used to tune the PID control systems on the robot. These track bars enable the adjustment of PID parameters, and the information is transmitted to the robot to remotely command the robot's control systems. Users can select a specific skill from the dropdown menu to modify the associated parameters. The save button is used to instruct the robot to save the current PID parameters onto its configuration file.

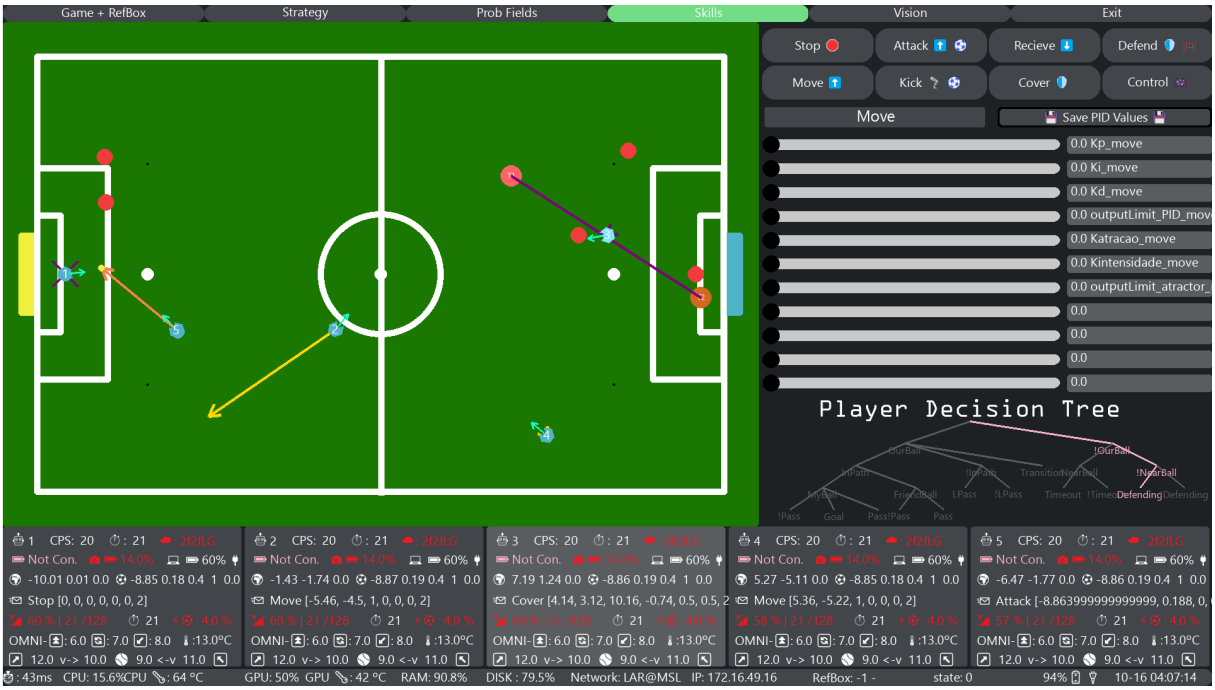


Figure 66: Skill tab with PID calibration trackbars, skill buttons and player decision tree

8.1.5 Vision

The "Vision" tab is a tool for the robot camera verification. It can simultaneously show camera images from all robots and select the specific image to show. The choice of camera is based on five dropdown menus, each corresponding to a robot. The image shown can be from the omni-vision camera, with or without the object detection and the Kinect depth camera either the RGB or Depth images, among others. Figure 67 has the images from robots number two, three and four. Robot number two requests the Kinect image with object detection while robot number three requests the omni-vision image with only the detected white lines. Lastly robot number four requests the image from the omni-vision with object detection.

The protocol used to send the images in real-time to the base station is the Google Remote Procedure Calls (gRPC)[1]. In the team scenario, it is mostly used to make sure that the neural network is working well and the robot's vision systems are working as intended.

[1] gRPC is a high-performance open-source remote procedure call (RPC) framework that uses protocol buffers and HTTP/2 for communication.

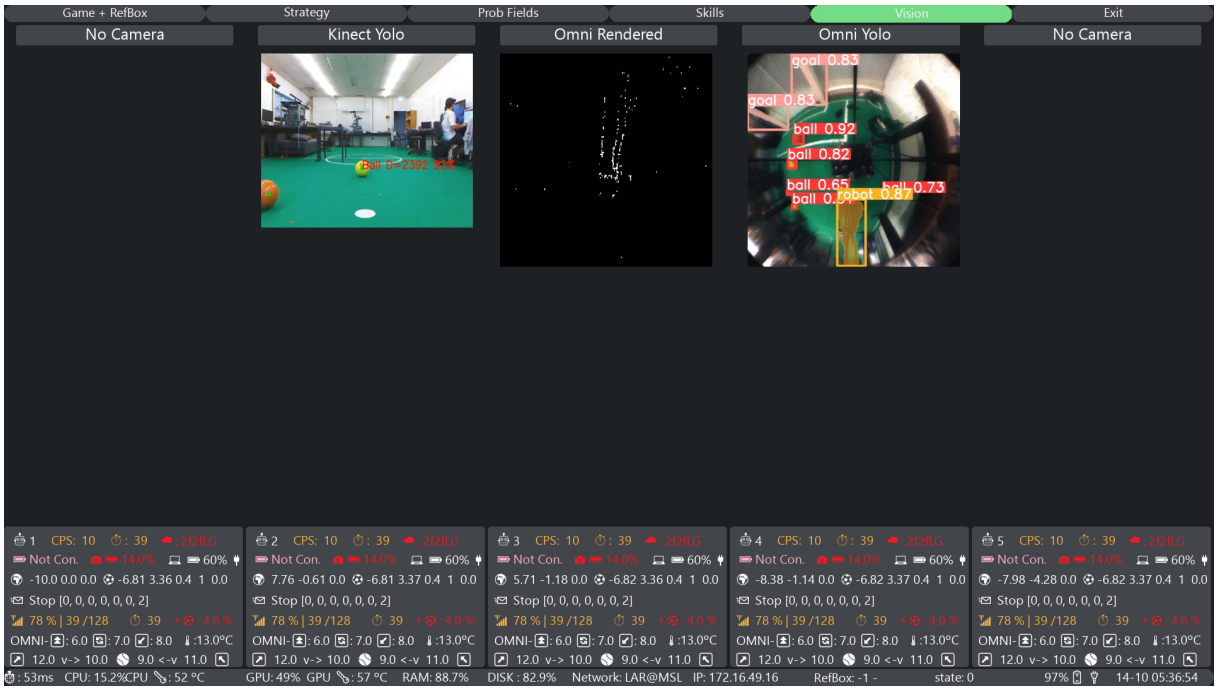


Figure 67: Vision tab

8.2 Graphical Representation

The base station graphical representations need to be as intuitive and interactive as possible. The reason is that the representation of values helps one to understand what the sent values mean. During a game, the values sent by the robots change multiple times a second, and its representation is the key to simplifying the interpretation of the values by humans.

8.2.1 Field and Robots

The highest screen percentage used is for the field and robots' representation. To draw the field, the size of every field line is required and it is available in the MSL rule book [3]. These guidelines still generate various possible field configurations, so the field drawn is always based on the same configuration structure. The field was designed to be scalable according to a scale factor as well as the screen resolution.

The robot's location and orientation are used to draw them onto the field. Based on the robot's location and orientation, the coordinates of each detected game element are calculated and drawn onto the field. The coordinates of the detected elements are relative to the robot. These are converted onto absolute coordinates before being represented. After drawing every robot and detected element, that same data is fed into the data fusion and the filtered locations are also drawn onto the field. Both the original and filtered information are shown and the way to distinguish both is based on the opacity. Each robot's ball

detection has a 50% opacity with the weighted average of the ball at 100% opacity. An example of this representation is visible in Figure 68.

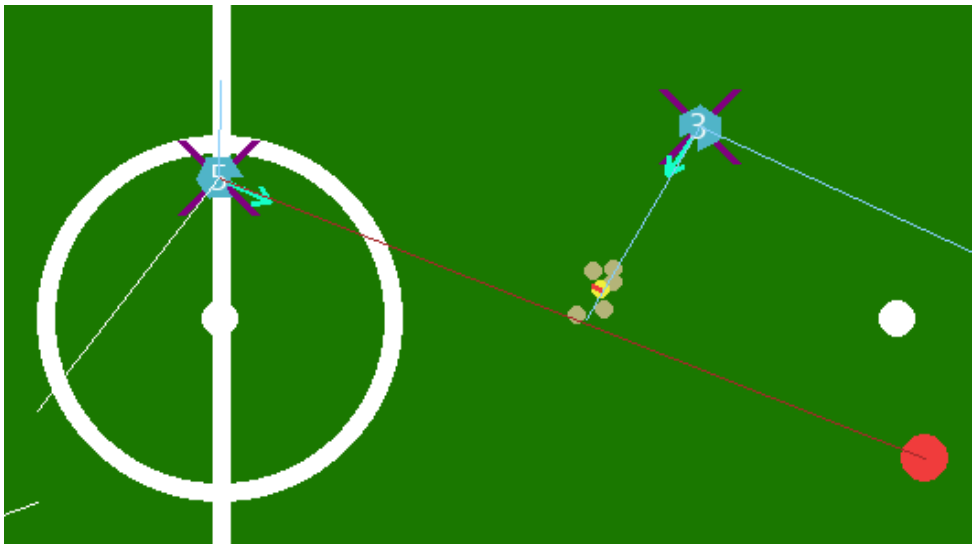


Figure 68: Every robot ball representation and weighted average

8.2.2 Robot Information

The robot information is a real-time table of all the important robot data. It provides information about sensor readings, battery charges, communications, robot states, and other relevant data. To make the information more intuitive and easy to interpret its representation is colour-coded. Specifically, white indicates normal operation, orange indicates a warning, red indicates an important alert and pink means something is not connected. Instead of words, emojis and icons are used to compile the information making it both intuitive and efficient in terms of screen space utilisation and human interpretation. Figure 69 shows an example of this data on Robot 3 where the Git[2] commit is outdated and, therefore, is in red while the camera battery is at 14%, which is low, and the capacitors of the kick are also at a very low charge of 4%. The battery is not connected so it is represented in pink. It is also visible that the computer battery is at 60% charge and is charging as shown by the plug icon next to the battery percentage.

[2] Git is a version control system that tracks code changes and allows developers to collaborate by committing code revisions.

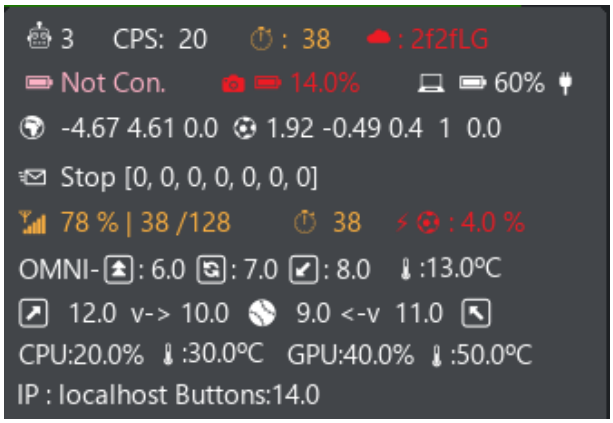


Figure 69: Robot information panel

8.2.3 Status Bar

A status bar is displayed at the bottom of the screen with the base station computer information. It is the same as the robot information squares but for the base station computer. It helps prevent memory management issues such as high CPU and GPU usage and temperatures, and it shows information about the iteration time of the base station code. On the right side is RefBox and network information as well as the date and time. The date and time are important to know not only for management reasons but for the log file name as well.

8.2.4 Skill Representation

Visual representation is also used in the skills because it is important to quickly perceive the robot's behaviour for fast human reaction as games are really fast-paced. Figure 70 shows the following representation:

- Stop Skill: Represented by a purple cross;
- Move Skill: Represented by a yellow arrow that points to the desired movement direction;
- Attack Skill: Represented with an orange arrow aimed at the ball;
- Kick Action: Represented by a blue arrow, specifying the target location for kicking;
- Pass Action: Represented by a blue arrow connecting the ball and the intended recipient player;
- Defend Skill: Represented by a brown circle surrounding the player;
- Control Skill: Not represented visually.

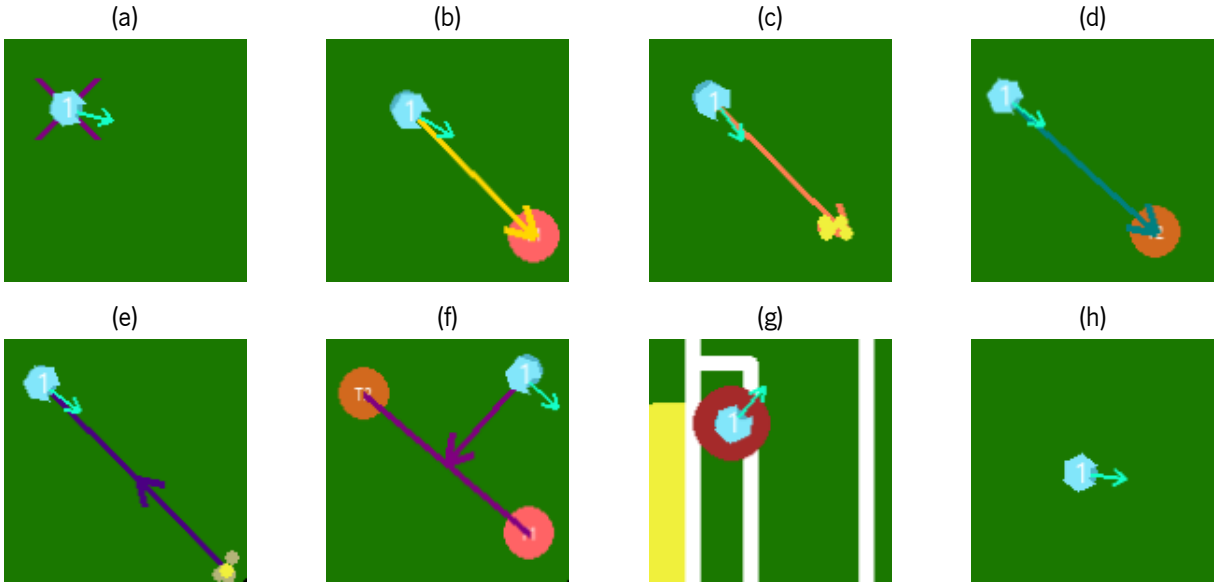


Figure 70: Base station skill representation: (a) Stop; (b) Move; (c) Attack; (d) Kick; (e) Receive; (f) Cover; (g) Defend; (h) Control

8.2.5 Heap Maps

The heat maps representation used within the heat map calibration tab, uses the matrix from the chosen robot and transforms the matrix of probabilities into a colour-coded image. The range of weights, from -1 to 1, is represented by colours transitioning from blue to red. The formula that calculates the RGB components and translates the weights of the matrix to a coloured matrix is shown in the equation 8.1.

$$\begin{cases} R = 128 + (127 * matrix[x][y]) \\ G = 0 \\ B = 127 - (127 * matrix[x][y])) \end{cases} \tag{8.1}$$

8.2.6 Keyboard Shortcuts

While the base station interface is fully operable with a standard mouse, certain keyboard shortcuts were also implemented to allow for more efficient controls. These shortcuts provide an alternative means of quickly accessing and using key features of the interface. The most useful and frequently utilised keyboard shortcuts are as follows:

Table 8: Base station keyboard shortcuts

Keys	Tab	Function
F1 - F5	All tabs	Switches between tabs
1 - 5	Strategy, Skills Heat-Maps	Selects the desired robot
q,w,e,r a,s,d,f	Skills	Chooses the skill to the selected robot
F12	All tabs	Resets the communication system
F6 or ESC	All tabs	Closes the program and saves the current log file

With these shortcuts complementing the mouse actions users can efficiently control all aspects of the simulation and interface. Some extra keyboard shortcuts exist for debugging and manipulation of the Webots environment.

8.2.7 Audio Feedback

With the large amount of real-time data displayed in the base station interface, additional audio cues were implemented to provide critical feedback for key events. These audible alerts utilised pygame audio playback functionality. The following notable robot events trigger warning sounds:

- Robot connecting/disconnecting from base station;
- Robot running outdated code version;
- Low robot/computer battery;
- Robot exhibiting slow internal loop time;
- High robot CPU/GPU usage/temperature.

Every audio command has the following format: Robot number X, warning command. For example "Robot number two low battery". These audio cues proved invaluable during debugging by quickly highlighting issues like low battery or performance problems. The alerts also prevented someone from working on outdated code, comparing the Git commits and versions by warning the user. During competitions, with many users interacting with the base station, the audible feedback enabled easy notification of important robot events.

Chapter 9

Tests and Results

The game strategy had different study approaches: in the simulator on specific plays, against a human-controlled team, and in the real-world. As expected, there were differences between the simulation and the real-world (RoboCup games). All the data acquired in the real-world was recorded in the RoboCup 2023 competition, where the LAR@MSL team played with the best teams in the world. Even with the transition between the simulation and the real-world and the competition environment, the game strategy was in play in all games and the results were clear.

9.1 Play Generator and Game Strategy Performance

The play generator was evaluated based on the underlying probability formulae and the processing time required. By analysing these timing metrics and the play probabilities produced by the generator, its effectiveness and speed were quantified. Its performance was benchmarked on a PC with the following specifications:

- CPU - Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz;
- RAM - 16 GB;
- GPU - AMD Radeon Pro 555X.

The results showed that constructing the play selection graph and executing the path-finding algorithm to pick the highest probability play took approximately one millisecond on average. On average, the base station had a loop time of about 35 milliseconds. The time not used by the play generator and path-finding algorithms were used by the heat map, positioning parts, and communication with sending and receiving the information. Another big part of the iteration time was allocated to the representation and user interface.

9.2 Communication System

The results presented in this section were acquired during the games played at RoboCup 2023. They are based on analysis of the log files generated by the base station during matches. These log files register all sent and received packets for every robot on the team. Several of the log files from the games were evaluated to calculate the performance of the communication system. Figure 71 shows the packet rate received by the base station from the robots. Eleven games were analysed, and the graph is divided between teams, games and robots. Each bar corresponds to the packet rate for a team robot in a specific game against one of the three teams presented.

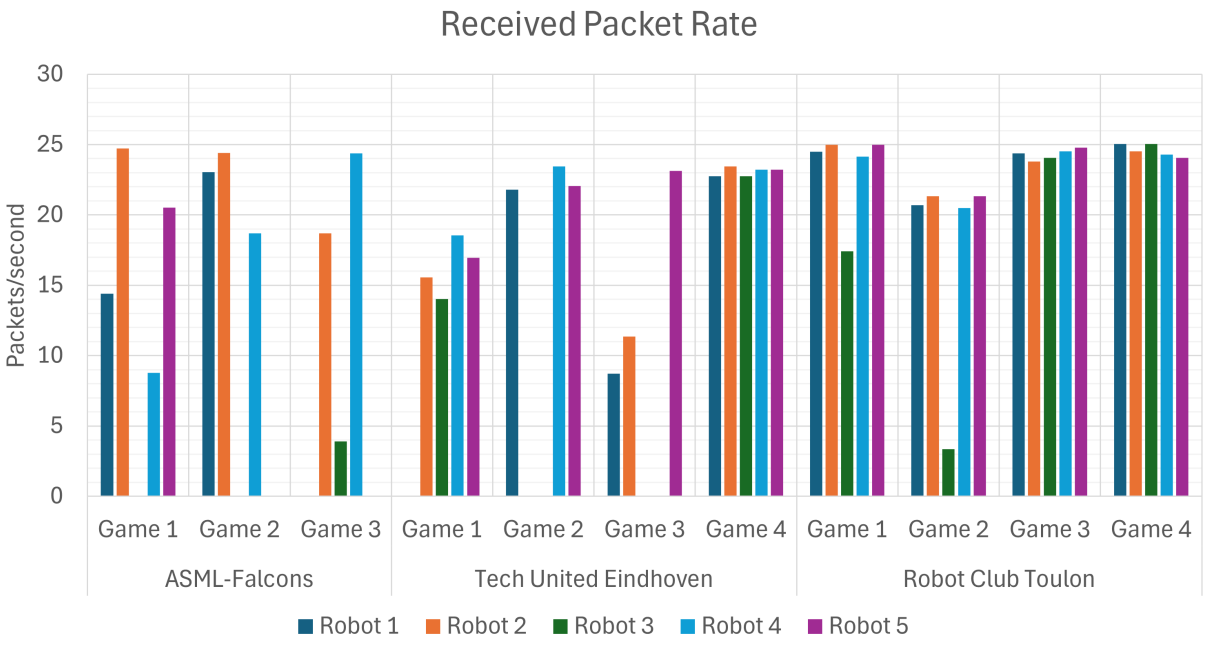


Figure 71: Packet rate of received messages by the base station in RoboCup 2023 games

In some games it is visible that there are not five bars for each of the five robots. When a robot had hardware problems, it did not enter the field to play and did not communicate with the base station. The team only played with three robots in games two and three against ASML-Falcons and games two and three against Tech United Eindhoven. The team had one damaged robot in game one against ASML-Falcons and in game one against Tech United Eindhoven. The similarity between the game numbers across different teams regarding the damaged robots is because these were played on the same day.

Occasionally, robots had hardware problems during the game when they were taken out for an amount of time and then went back in. This is visible in Figure 71 in games where the received packet rate is low on a specific robot. For example, games one and three against ASML-Falcons, game three against Tech United Eindhoven and game two against Robot Club Toulon.

On average, the communication system had a success rate of over 80% even when counting the mid-game damaged robots. It is visible increased stability in the communication system against Robot Club Toulon, and this is due to the fact that they use the 2.4 GHz frequency instead of the 5 GHz as the other teams do.

To summarise, communication reliability is an influential 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, having at least one packet per second is advisable. Should the loss be higher, robots would continue playing but with decreased performance and stability. An alternative play is generated and given to the robots to counteract the packet loss effects to maintain their autonomous behaviour. This proves the reliability of the UDP sockets used by the game strategy in highly saturated environments. Significant packet loss was never observed in a non-saturated environment as are the conditions in the LAR facilities.

9.3 Heat Maps and Calibration

The position calculation is carried out every iteration in all robots. The first map shown in Figure 72 is an offensive situation where robot number 5 will receive a pass from robot 2. Its main objective is getting close to its teammates to increase the pass success probability. The green lines represent the movement intention of every robot. Robot 3 will move to the centre of the field to avoid staying behind the opponent and to get a clean line of a pass from Robot 5.

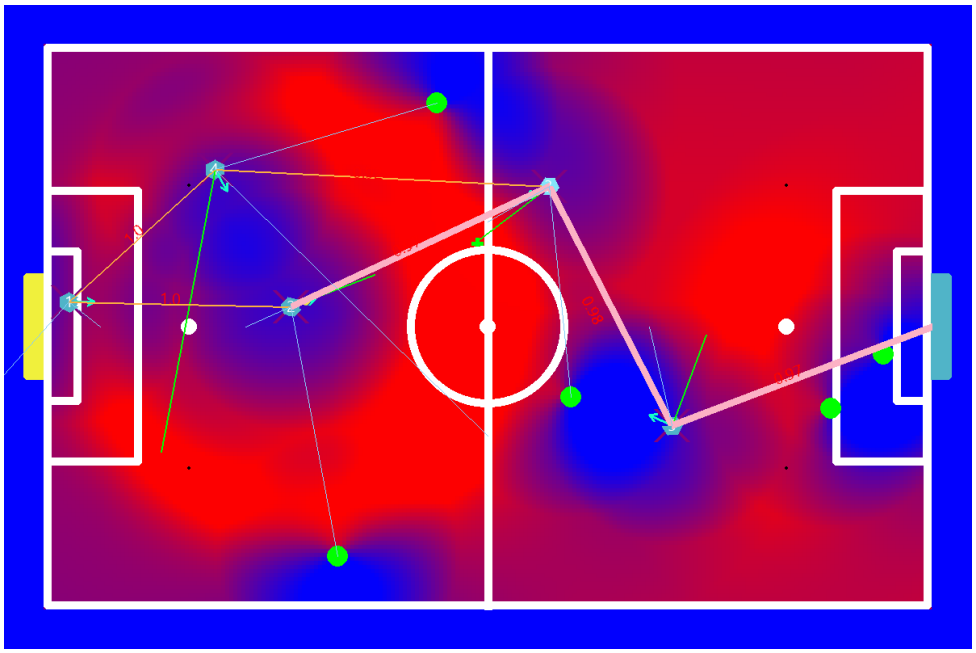


Figure 72: Offensive probability map

The second map, Figure 73, shows a defensive situation where the robots intend to move to positions to cover one or more opponents. The map shows robot number five intention, where the zone and distance are the elements with more influence. This occurs because, in this situation, robot number 2 which is a teammate is the closest to the ball. Although this is a defensive scenario, there is no immediate threat from the opposing team. Therefore, the best strategy is to maintain a defensive position, keeping a suitable distance from the opponent to be able to receive a pass but remaining close enough to react quickly if the opposing team gains possession of the ball.

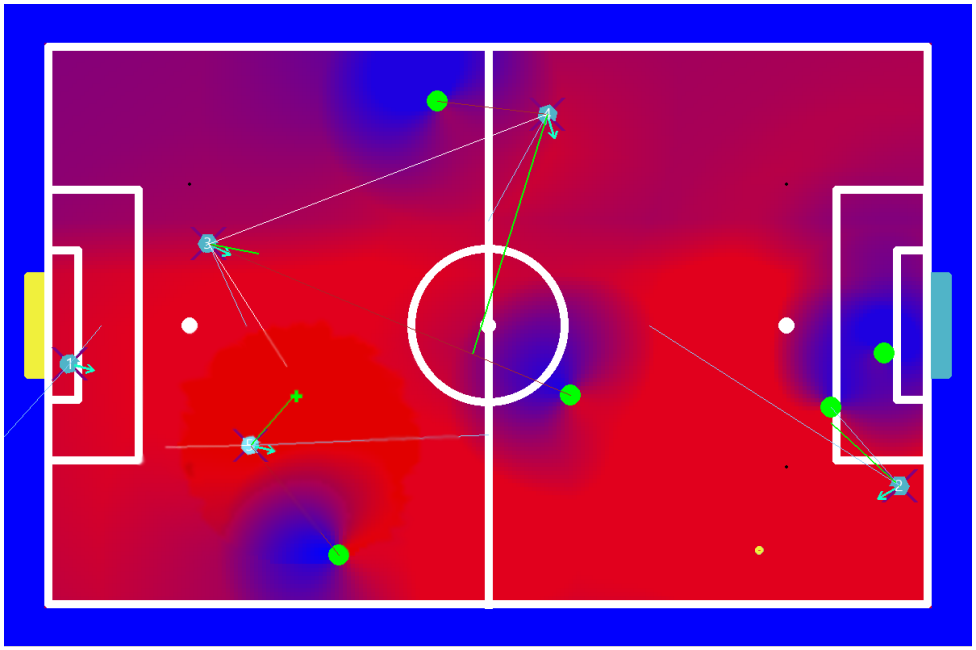


Figure 73: Defensive probability map

The influences of all maps have a specific weight associated with them and these need to be adjusted according to the desired output of a particular game and player situation. For an output of the decision trees, the desired maps' weight 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 behaviours 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, which is repeated until the robot's behaviour 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.

9.4 Simulator

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

9.4.1 Case 1

The first game situation, presented in Figure 74, 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 positions itself near the ideal goal distance, giving an alternative to 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 optimise each other.

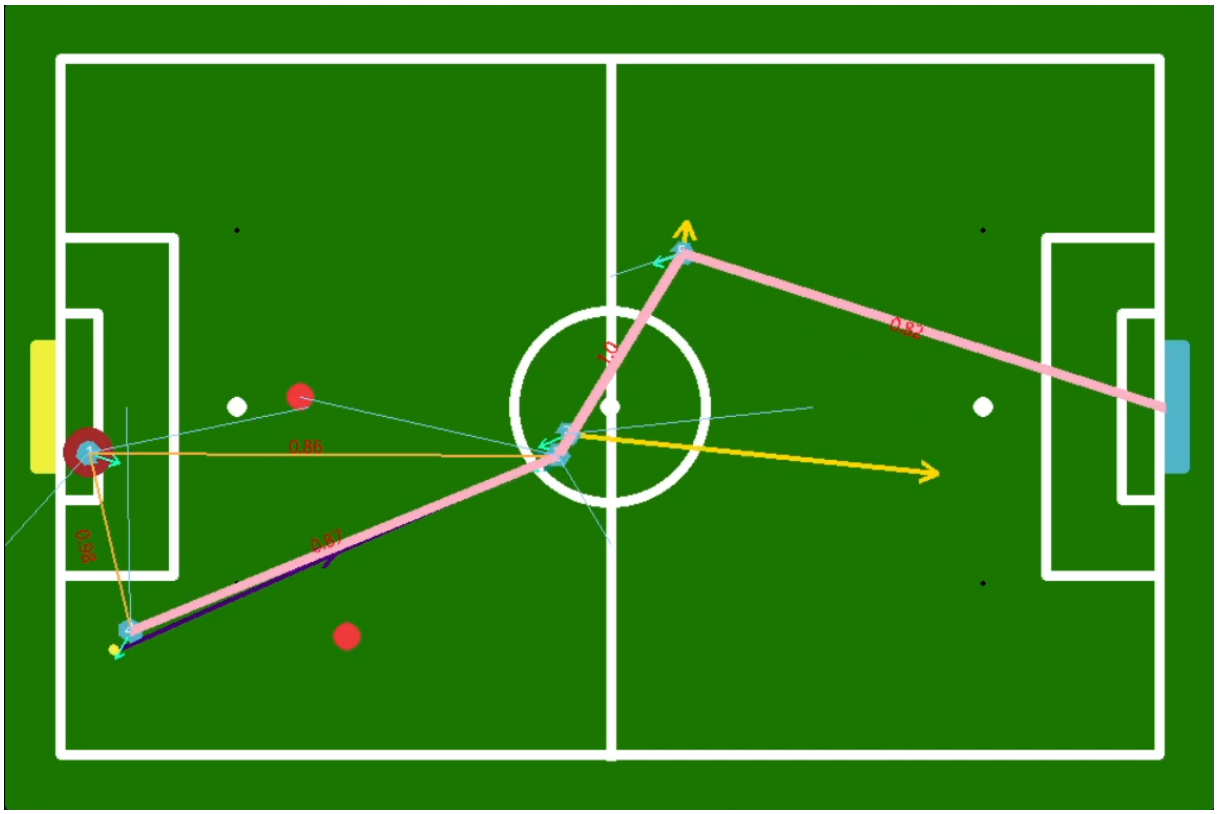
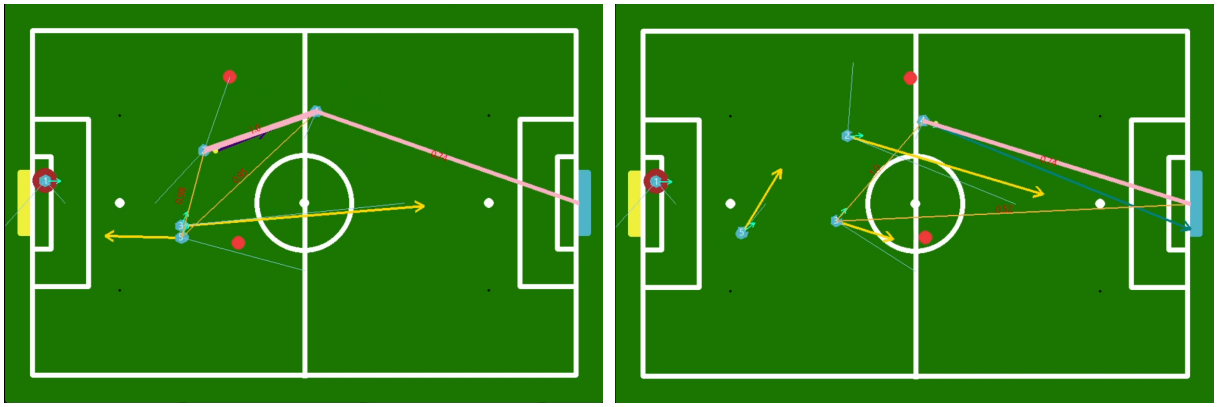


Figure 74: First game situation in the simulator

9.4.2 Case 2

Figure 75 has two images from the same game scenario. Figure (b) is the situation after the first pass. In Figure 75 (a), 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. 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 2 and 4 is made, robot number 5 positions itself more to the centre, and robots 3 and 2 change tasks. This happened because robot number 2 was closer to the desired spot. Since no other opponents are detected, robot number 4 also has a relatively high probability of scoring, that being the assigned task.



(a) (b)

Figure 75: Second game situation in the simulator: (a) Initial situation; (b) Situation after the first pass

9.4.3 Case 3

Another offensive play is visible in Figure 76, 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. In contrast, 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.

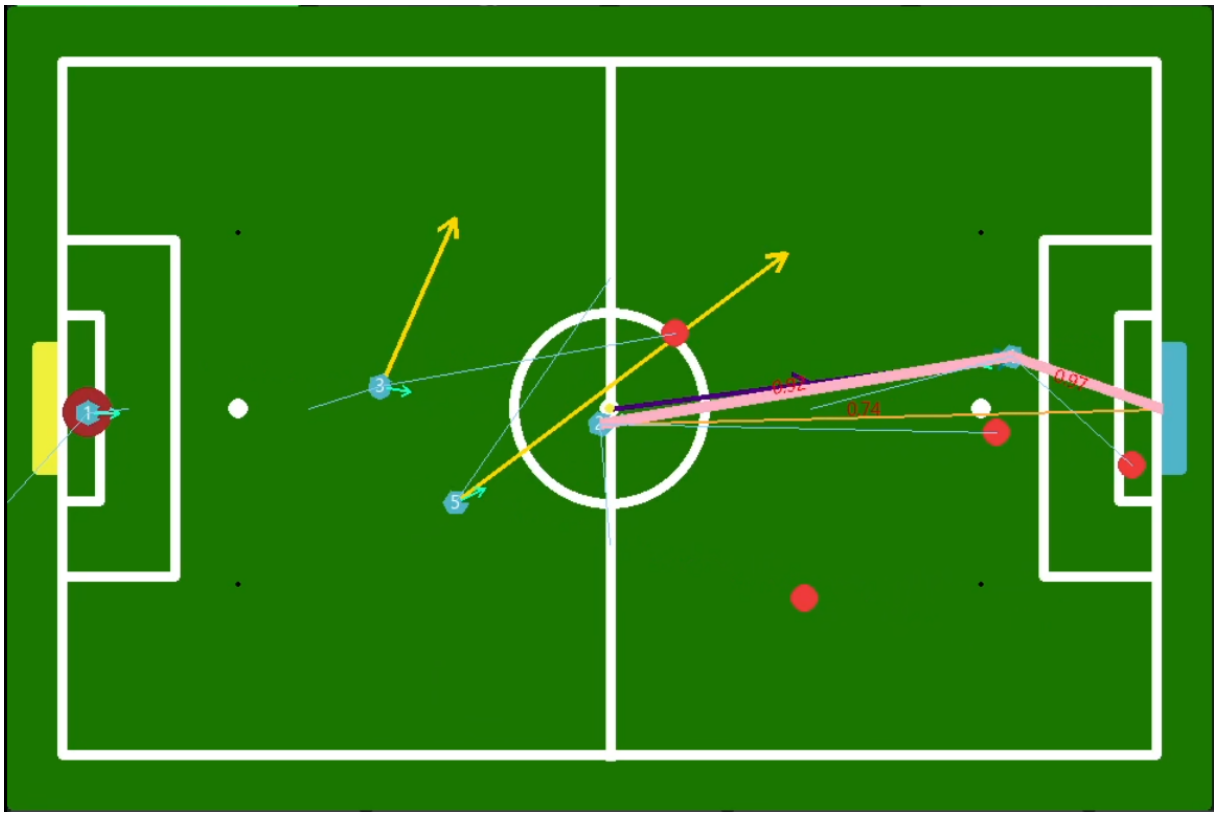


Figure 76: Third game situation in the simulator

9.4.4 Case 4

Now, shown in Figure 77, it 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. To maintain high pressure, the robot closest to the ball still tries to attack it and 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 the pass to the opponent near the centre.

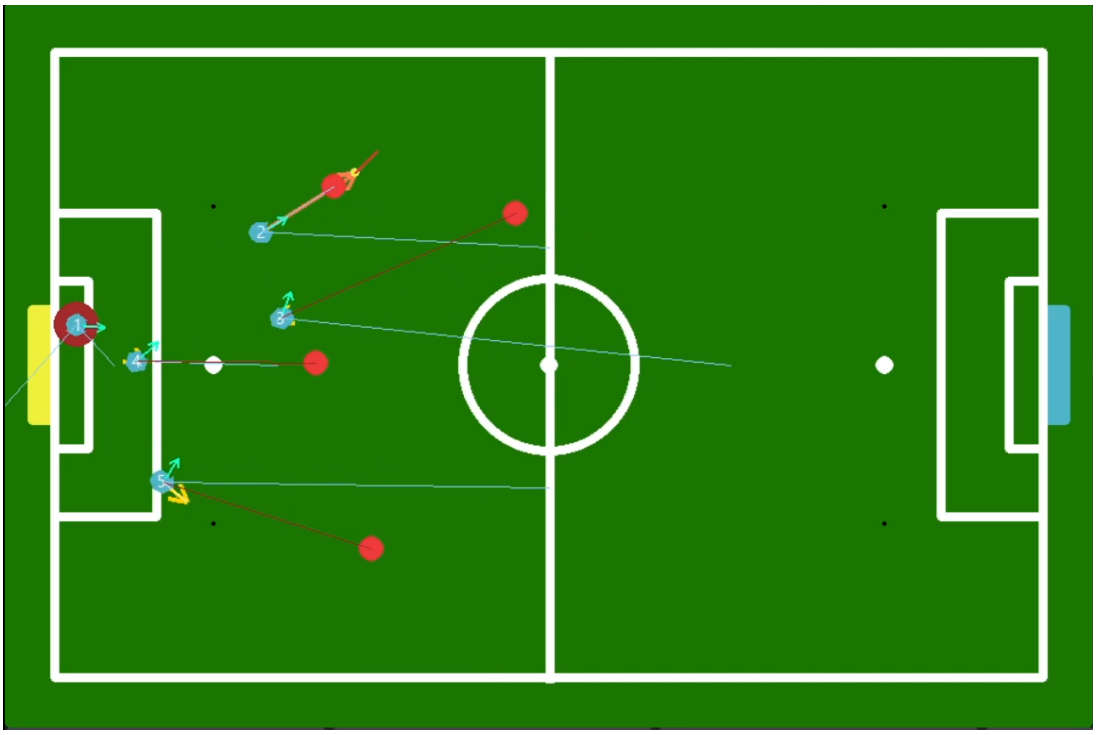


Figure 77: Fourth game situation in the simulator

9.5 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. Like in common football computer games such as the FIFA video game, a game controller was given to each human to control one of the opponent robots, visible in Figure 78. 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. A video of the strategy development is available on LAR's Youtube channel [54], where the simulator, the base station, and the game strategy actions sent to the robots are shown. The strategy proved to be more synchronised than the human team and their movements were visibly more coordinated.

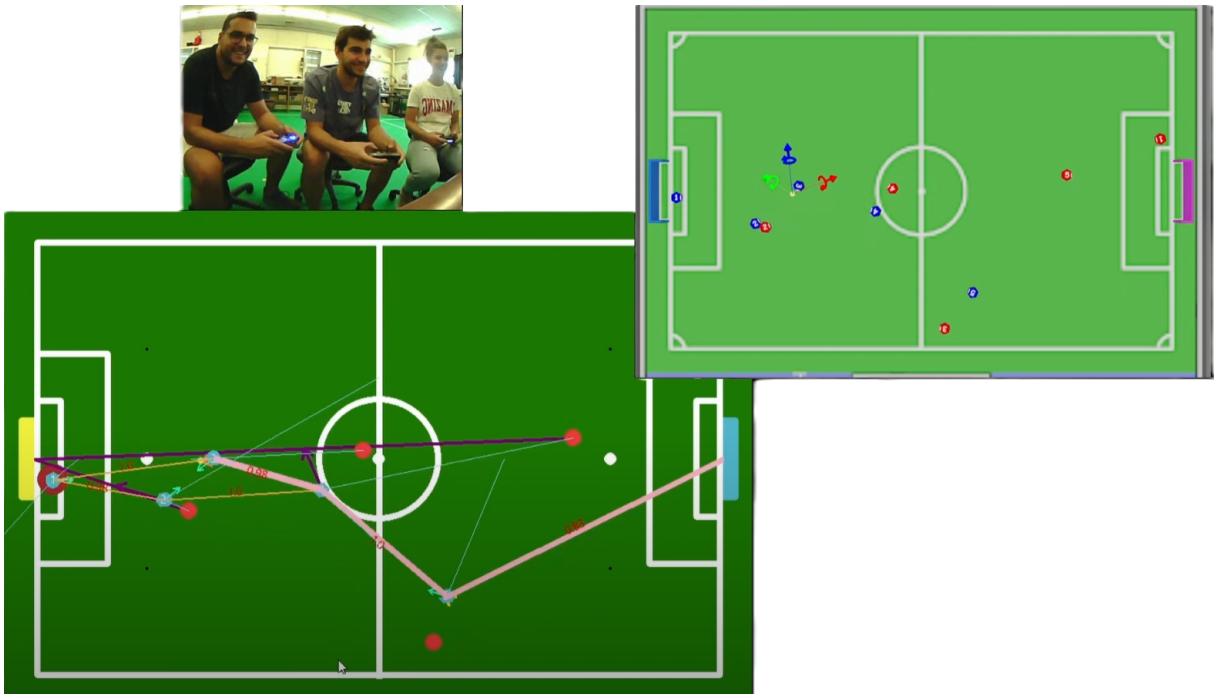


Figure 78: Game strategy against humans in the simulator

It effectively cut the lines of pass and intercepted the ball in a pass. In the use of the Cover skills, it is also perceptible that the Hungarian algorithm begins to associate each robot with an opponent to defend, where the associated opponent is always the nearest to that robot. It is also perceptible that as soon as the robots get ball possession, the time required to play and kick to the goal is also very little. During offensive situations, the position of the robots ahead of the ball to create alternative lines of the pass is also visible and effective. Even though they were not necessary, alternative lines of pass constantly exist.

9.6 Competition Games

The transition to the real-world was studied on different occasions but real 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 on LAR's Youtube channel [55]. It is visible in the real-world image (filmed by one of the competition teams) and in the base station information gathered by the robots and what were the decisions taken. All the games were recorded during the competition, 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.

9.6.1 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. Due to the poor robot control system, it was not able to receive the ball from robot 2. It is also perceptible in the information update presented how inconsistent the communication-saturated environment was. The flicker of each robot position and opponent detected was due to the accuracy of the localisation and detection systems.

9.6.2 Play 2

This shows a defensive situation, where all the robots positioned themselves 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. The information gathered by the strikers about the ball location was extremely important to the goalkeeper, who, based on that, could act accordingly.

9.6.3 Play 3

In play 3, a defensive situation, a synchronisation 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. This play proves the effectiveness of the Cover skill.

9.6.4 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 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.

9.6.5 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 game 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.

9.7 STP vs. PBS

Since the game 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.

9.7.1 Differences

Both strategy algorithms are based on skills that the robots can perform, but how these are attributed to the robots is very different. While the STP architecture is decentralised and the synchronism is maintained, the PBS is centralised 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 [13, 12]. One of the problems is that the plays are thought to have a static number of players. If the number of players in a team changes, the playbook needs to be rebuilt. The PBS developed skips the need to implement tactics, making it simpler, easier, and less time-consuming to implement.

9.7.2 Limitations

PBS also has some disadvantages, and the limitations are almost all based on information, actions, and the opponent's behaviour. Firstly, since the decision-making is centralised, the latency of communications can be a problem. Therefore, the communication system must be as fast and reliable as possible. It also depends on the robots' information, 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, should the robots not behave well given a certain skill, the game's outcome can be

ruined even though the game strategy still tries to give the best output possible. The game strategy is as efficient as the robots, which can 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, achieving a stable solution can take quite some time. After the first stable solution, adapting and changing the behaviour is quicker and more straightforward.

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

9.7.3 Modularity, Flexibility and Scalability

The game strategy's modularity, flexibility, and scalability were all factors considered while designing the PBS algorithm. Concerning modularity, the system is easy to change and adapt to different game situations and opponents' behaviour. By adjusting the heat map weights and the opponent's influence in the pass and goal probability formulae, the robot's risk 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 also easily scalable since it does not rely on the number of players on either team.

9.8 Summary

The game strategy demonstrated promising performance in tests across different environments, including the simulator, games against humans and real-world competition. In the simulator, the strategy could direct robots to make appropriate positioning and passing decisions in varied offensive and defensive scenarios. The calibration process allowed tuning of the influence maps to achieve the desired robot behaviours. Games against human opponents provided valuable insights into dynamic situations requiring strategy adjustments.

The real-world competition enabled the evaluation of the complete system, including communication, perception, and control. The recorded log files and video footage documented good strategy execution given the constraints of real-world conditions. Challenges like communication delays, detection errors, and hardware failures impacted in performance but did not prevent the game strategy from making the right decisions for every game situation. The testing showed the viability of the probabilistic play selection, influence maps, and role assignment logic despite imperfect real-world execution. Further refinements to the algorithms and robot systems can be built on these initial results.

Chapter 10

Conclusions

The conclusions chapter summarises the key findings and takeaways from the research presented in this dissertation. A new decision-making method regarding the game strategy of multi-agent autonomous robots can significantly improve football strategies and robotics in general. The game strategy architecture presented solves a robotics problem without a typical human solution but with a mathematical and robotics approach.

The tools created to accelerate the development time and to enable the team to play and participate in competitions proved efficient and crucial to this dissertation. Developed and used throughout this dissertation, the base station was the central tool that handled communications, debugging, representation, robot interaction and game strategy.

The Probability-Based Strategy (PBS) proved modular, flexible, and easily adaptable to different game situations with minimal effort. It allows different behaviours 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 a new game strategy when compared with STP. The time needed to adjust any parameter is also negligible. Another significant advantage is that this method eliminates the need for plays and a playbook, which are usually very time-consuming tasks.

Simulating the environment was also a key factor in accelerating the development, testing, and even calibrating the game 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.

The work developed in this dissertation relies on a reasonably reliable communication system. In some saturated environments, the network's reliability can become compromised. Even so, the stability of the communication system developed proved to be very stable, in hard conditions such as in RoboCup, where thousands of wireless devices are used.

The participation in competitions allowed the team and, more specifically this dissertation, to acquire results in the following scenarios:

- Manually positioning an opponent;
- Strategy vs. manually controlled by the base station;
- Strategy vs. humans controlling the opponents with game controllers;
- Strategy used in RoboCup 2023.

The culmination of all the teamwork as well as this dissertation, achieved 4th place in RoboCup2023. Besides the competition participation, a paper was published [56], under the name "Probability-Based Strategy for a Football Multi-Agent Autonomous Robot System", in the Q1 journal Robotics, on volume 13 under the special issue "Multi-Robot Systems: State of the Art and Future Progress".

10.1 Prospect for Future Work

This dissertation is a big part of the LAR@MSL project and is the starting point for any future development of game strategy, decision-making and probability-based decisions, heat-maps and robot football. Some implementations are planned for the future, as follows:

- New decision tree method based on a trainable neural network;
- To study eventual new action probability formulae based on real-world hardware feedback;
- New approach for the data fusion methods;
- Study of new algorithms to be applied in the heat map positioning;
- Apply the ideal path probability in defensive situations;
- Optimise the graph generation.

Bibliography

[1] Nardi, D.; Noda, I.; Ribeiro, F.; Stone, P.; von Stryk, O.; Veloso, M. RoboCup Soccer Leagues. *AI Mag.* 2014, *35*, 77–85. Available online: http://dx.doi.org/10.1609/aimag.v35i3.2549 (accessed on 6 August 2023).

[2] Stone, P. Will Robots Triumph Over World Cup Winners by 2050? Available online: https://spectrum.ieee.org/robocup-robot-soccer (accessed on 18 July 2023).

[3] 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 2 June 2023).

[4] Ribeiro, Hélder Ricardo F.: Estratégia de coordenação e cooperação de sistemas Multi-Agente, University of Minho, Master's Thesis, 2016. – Available online: https://hdl.handle.net/1822/65415.

[5] Silva, Pedro Henrique Meneses Osório G.: Simulador 3D multi-plataforma para competições robóticas, University of Minho, Master's Thesis, 2018. – Available online: https://hdl.handle.net/1822/59470.

[6] Silva, Sérgio Filipe V.: Controlo e monitorização da equipa de robôs futebolistas MINHO TEAM, University of Minho, Master's Thesis, 2020. – Available online: https://hdl.handle.net/1822/65422.

[7] 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 24 March 2023).

[8] RoboCup Objective - Pushing the State-Of-The-Art. – Available online: https://www.robocup.org/objective (accessed on 13 March 2023).

[9] 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.

[10] 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. Available online: https://doi.org/10.1007/978-3-319-29339-4_9.

[11] 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.

[12] 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. Available online: https://doi.org/10.1243/095965105X9470.

[13] 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. Available online: https://doi.org/10.1007/978-3-030-00308-1_23.

[14] TechUnited: Tech United Eindhoven Team Description 2023. Available online: https://www.techunited.nl/uploads/Voetbalrobots/Kwalificatie/23-02-28-Tech_United_TDP_2023.pdf (accessed on 11 April 2023).

[15] TechUnited ; TU/e: THE MESSI OF ROBOT SOCCER IS A TRICYCLE WITH A FISHEYE. Available online: https://www.tue.nl/en/news-and-events/features/soccer-robots (accessed on 12 April 2023).

[16] 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. Available online: https://doi.org/10.1007/978-3-030-27544-0_7.

[17] Orr, J.; Dutta, A. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* 2023, *23*, 3625. Available online: https://doi.org/10.3390/s23073625.

[18] Antonioni, E.; Suriani, V.; Riccio, F.; Nardi, D. Game Strategies for Physical Robot Soccer Players: A Survey. *IEEE Trans. Games* 2021, *13*, 342–357. Available online: https://doi.org/10.1109/TG.2021.3075065.

[19] 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. Available online: https://doi.org/10.1007/978-3-319-07776-5_26.

[20] 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, Available online: https://doi.org/10.1109/BRACIS.2018.00081.

[21] Budanov, D.; Feltracco, J.; Kamat, J.; Medrano, R.; Naeem, S.; Neiger, J.; Osawa, R.; Pan, M.; Peterson, E.; Shaw, A.; Stuckey, W.; Ting, J.; Woodward, M. RoboJackets 2017 Team Description Paper. Available online: https://ssl.robocup.org/wp-content/uploads/2019/01/2017_TDP_RoboJackets.pdf (accessed on 18 November 2023).

[22] 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. Available online: https://doi.org/10.1145/3461353.3461360.

[23] 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.

[24] 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. Available online: https://doi.org/10.1145/3071178.3071194.

[25] 5dpo-2000 Team Description for Year 2003", António Paulo Moreira, Paulo Costa, Armando Sousa, Luís Paulo Reis, Team Descriptions of the RoboCup 2003 Games and Conferences.

[26] RoboCup Middle Size League. – Available online: https://msl.robocup.org/ (accessed on 21 August 2023).

[27] Turtle 3. – Available online: https://gitlab.tue.nl/tech-united-eindhoven/Turtle3 (accessed on 5 August 2023).

[28] Soccer Robots. – Available online: https://www.techunited.nl/?page_id=1962 (accessed on 5 August 2023).

[29] Coordinating distributed autonomous agents with a real-time database: The CAMBADA project. Available online: http://robotica.ua.pt/CAMBADA/docs/pub2004/almeida.04a.pdf (accessed on 5 August 2023).

[30] Team, Water: Water Team Description 2019. Available online: https://msl.robocup.org/wp-content/uploads/2019/02/Water-TD-2019.pdf (accessed on 7 April 2023).

[31] CAMBADA'2017: Team Description Paper. Available online: http://robotica.ua.pt/CAMBADA/docs/qualif2017/CAMBADA-tdp-2017.pdf (accessed on 5 August 2023).

[32] Almeida, Luís & Azevedo, José & C. Bartolomeu, Paulo & Brito, E. & Cunha, Bernardo & Figueiredo, João & Fonseca, Pedro & Lima, C. & Marau, Ricardo & Lau, Nuno & Pedreiras, Paulo & Pereira, Artur. CAMBADA: Team Description Paper.

[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.

[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.

[35] 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).

[36] Log Probabilities. Available online: https://chrispiech.github.io/probabilityForComputerScientists/en/part1/log_probabilities/ (accessed on 8 September 2023).

[37] O'Hagan, A. Bayesian Statistics: Principles and Benefits. Available online: https://edepot.wur.nl/134085 (accessed on 7 September 2023).

[38] Wang, Y.; Shenhan, J.; Chen, Z.; Huang, Z.; Xiong, R. Multi-Agent Collaboration for Feasible Collaborative Behavior Construction and Evaluation. 2019. Available online: Available online: https://www.researchgate.net/publication/336146890_Multi-agent_Collaboration_for_Feasible_Collaborative_Behavior_Construction_and_Evaluation.

[39] Quinlan, J.R. Induction of Decision Trees. Available online: https://hunch.net/~coms-4771/quinlan.pdf (accessed on 13 October 2023).

[40] Song, Y.Y.; Lu, Y. Decision tree methods: applications for classification and prediction. *Shanghai Arch. Psychiatry* 2015, *27*, 130–135. Available online: https://doi.org/10.11919/j.issn.1002-0829.215044.

[41] 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. Available online: https://doi.org/10.37418/amsj.10.3.29.

[42] 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. Available online: http://dx.doi.org/10.1108/IR-02-2016-0069.

[43] Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* 1995, *2*, 83–97. Available online: https://doi.org/10.1002/nav.3800020109.

[44] 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.

[45] Lopes, Gil and Ribeiro, A. Fernando and Pereira, Nino, "Catadioptric system optimisation for omni-directional Robocup MSL robots", 2012, Available online: https://hdl.handle.net/1822/21158.

[46] Webots Reference Manual R2023b: Supervisor. Available online: https://cyberbotics.com/doc/reference/supervisor (accessed on 20 October 2023).

[47] 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).

[48] 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.

[49] Ribeiro, F.; Moutinho, I.; Pereira, N.; Oliveira, F.; Fernandes, J.; Peixoto, N.; Salgado, A. High accuracy navigation in unknown environment using adaptive control. Volume 5001, pp. 312–319, ISSN: 0302-9743. Available online: http://dx.doi.org/10.1007/978-3-540-68847-1_30.

[50] Barrett, P.; Hunter, J.; Miller, J.T.; Hsu, J.-C.; Greenfield, P. Matplotlib— A Portable Python Plotting Package. 2005. Available online: https://www.researchgate.net/publication/234238535_matplotlib_--_A_Portable_Python_Plotting_Package.

[51] FIFA Futsal Coaching Manual. Available online: https://cdn1.sportngin.com/attachments/document/000a-2348966/FIFA_futsal-coaching-manual.pdf (accessed on 6 October 2023).

[52] "NumPy Documentation" NumPy documentation - NumPy v2.0.dev0 Manual. https://numpy.org/devdocs/ (accessed on 27 January 2024).

[53] "Documentation" OpenCV https://docs.opencv.org/4.x/ (accessed on January 3 2024).

[54] "LAR@MSL Strategy: Humans vs. Robots in the Simulator" - https://www.youtube.com/watch?v=SVYgXKeGaf0.

[55] "LAR@MSL Strategy: RoboCup Plays" - https://www.youtube.com/watch?v=63nJXTCfQEk.

[56] 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. Available online: https://doi.org/10.3390/robotics13010005.

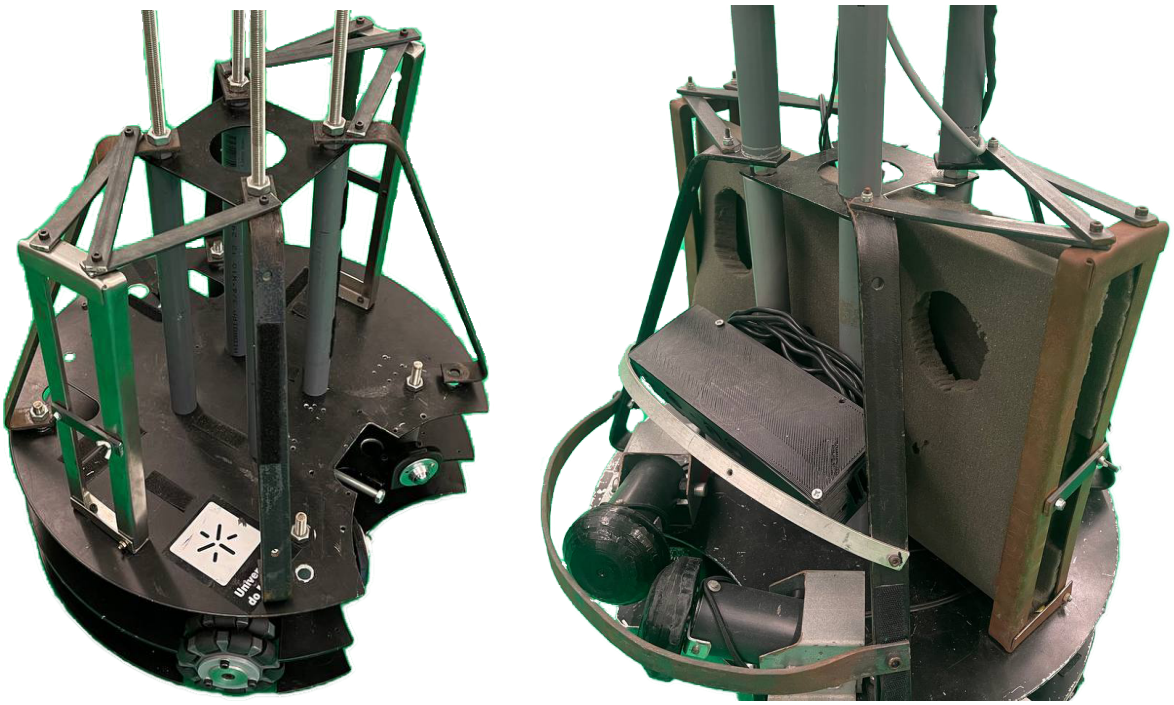
Appendix A
Extra Work

While the core focus of the dissertation involves high-level strategy, extra work was equally crucial to allow the robots actually to play football. Ensuring the hardware, electronics and low-level software functioned properly, the game strategy algorithms was a requirement for obtaining usable real-world results. Tasks such as designing the robots' chassis, building batteries, making the robot's sportswear and solving other hardware and electronics problems do not directly connect to the dissertation's main goals but were essential prerequisites. Without this critical "extra work" to handle the underlying robot capabilities, it would not be possible to participate in the "Festival Nacional de Robótica" and RoboCup 2023 to effectively evaluate the strategy in a real-world scenario.

Computer Protection Chassis

Each robot is equipped with a laptop computer enclosed in a protective casing. An aluminium and steel cage was installed on top of each robot to shield the computer from damage. This sturdy metal cage prevents any direct impacts from other robots or objects from reaching the laptop. The cage is mounted to the robot's aluminium frame as well as the robot's four main screws. Figure 79 shows the steel cage mounted onto the aluminium frame (a) and the protective computer sponge (b).

Additionally, a sponge material surrounds the laptop inside the cage to absorb the collisions. The sponge has two holes to allow proper airflow and cooling of the computer. With this protective enclosure, the vital computers are safeguarded from harm if the robots encounter collisions, as they sometimes do.



(a) (b)

Figure 79: Laptop computer protection cage: (a) Steel frame; (b) Protective sponge

Robot Sportswear

As predefined in the rules, every robot needs to have a team identification colour and respective robot number. Sportswear T-shirts were made for every robot in two colours blue and red. Every team needs to have two distinct colours. The only requirement is that the colours presented are saturated. Cooling was considered and so on the sides of every shirt, there are holes to facilitate the laptop airflow. The LAR@MSL robot sportswear is visible in Figure 80.

(a) (b) (c)

Figure 80: Robot sportswear: (a) Front view; (b) Side view; (c) Back view

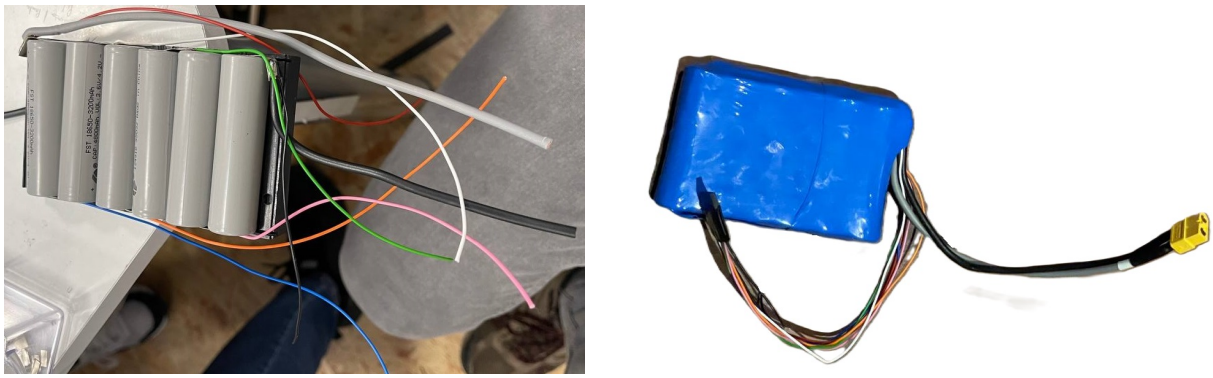
Batteries

The robots are powered by lithium-ion batteries with a voltage of 24 V that were built by the team due to budget constraints. Specifically, the batteries utilise 18650 lithium-ion cells, each containing twelve cells configured six in series and two in parallel. The individual 18650 cells have a rated capacity of 3200 mAh, so each full robot battery has a total capacity of 6400 mAh. The maximum continuous discharge rate for the 18650 cells is 0.5 C, which equates to 3.2 A continuous draw per battery. For a short duration, the cells can handle pulse discharges up to 3.25 C or 20.8 A per battery. With this battery configuration and the capabilities of the 18650 cells, each robot can operate for extended periods within the continuous current limit while still having the pulse power capacity required for peak demands. The parallel grouping of the cells allows them to deliver sufficient current. The 18650 lithium-ion cells in each robot battery are connected using magnesium strips that were soldered together via spot welding visible in Figure 81 (a) and (b).

(a) (b)

Figure 81: Battery soldering process: (a) Spot welding; (b) Magnesium strips

The completed batteries do not contain internal Battery Management Systems (BMS). However, connectors were added to each individual cell to allow for cell balancing during charging, Figure 82 (a). The battery charging stations contain the necessary BMS circuitry to charge the batteries while monitoring and balancing each cell. Once assembled, the batteries are covered in heat-shrink tubing to protect the cells and connections. This heat shrink tubing insulates and isolates the battery while also providing a clean, streamlined exterior, Figure 82 (b).



(a) (b)

Figure 82: Battery elements: (a) Battery Management Systems (BMS) connector; (b) Protective heat shrink tube

To further protect and isolate the robot batteries, custom 3D printed cases were created for each one. These cases are numbered to correspond with their respective robots, visible in Figure 83 (a). Additionally, the cases each contain a small display screen that shows the current charge percentage and voltage of the battery, Figure 83 (b). This allows the battery status to be monitored. The cases also have a rail built into them, with the robots containing the inverse rail, Figure 83 (c). This rail system enables the batteries to seamlessly slide into place when inserted into the robots, preventing any backward connections. It also locks the batteries firmly in position during operation, preventing hazardous movement inside the robot body.

(a) (b) (c)

Figure 83: Battery 3D printed parts: (a) Case with robot number; (b) Monitoring screen; (c) Sliding rails