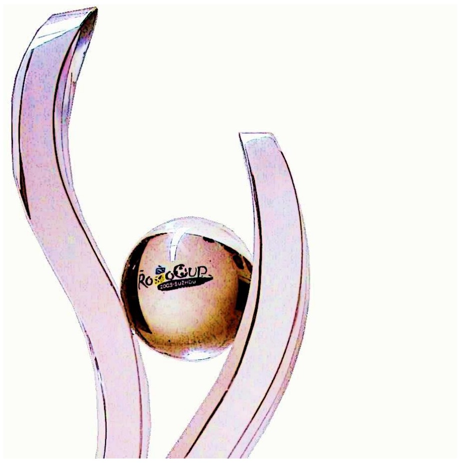
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2008 Informática

Nelson André
Saldanha Filipe

Funcionalidades de Decisão Individual e Coordenação na equipa CAMBADA

Individual and Coordinated Decision for the CAMBADA team



Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2008 Informática

Nelson André Saldanha Filipe

Funcionalidades de Decisão Individual e Coordenação na equipa CAMBADA

Individual and Coordinated Decision for the CAMBADA team

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2008

Nelson André Saldanha Filipe

Funcionalidades de Decisão Individual e Coordenação na equipa CAMBADA

Individual and Coordinated Decision for the CAMBADA team

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Nuno Panelas Nunes Lau, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Luís Filipe de Seabra Lopes, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente	**Doutor Tomás António Mendes Oliveira e Silva** Professor associado da Universidade de Aveiro.
	Doutor Luis Paulo Gonçalves dos Reis Professor auxiliar da Faculdade de Engenharia da Universidade do Porto
	Doutor José Nuno Panelas Nunes Lau Professor auxiliar da Universidade de Aveiro
	Doutor Luís Filipe de Seabra Lopes Professor auxiliar da Universidade de Aveiro

agradecimentos

Ao meu irmão e aos meus pais por estarem sempre lá. Aos meus amigos Pac, Daniel, Monstro e Vi por acreditarem em mim e no que faço. Aos meus orientadores Professores Nuno Lau e Luís Seabra Lopes pela preciosa ajuda e disponibilidade. Ao Gustavo Corrente por toda a dedicação ao projecto CAMBADA, pelo apoio e pelas noitadas a fazer código.

A todos os que de uma forma ou outra me apoiaram neste projecto.

A todos os elementos da equipa CAMBADA, pois sem o contributo de todos não teríamos conseguido o título...

palavras-chave

Sistemas multi-robô, RoboCup, comportamentos cooperativos, coordenação entre robôs

resumo

A coordenação em sistemas multi-robô é um aspecto crucial no futebol robótico. A maneira como cada equipa coordena cada um dos seus robôs em acções cooperativas define a base da sua estratégia.
Este trabalho tem como foco o desenvolvimento da coordenação e estratégia da equipa CAMBADA. CAMBADA é a equipa de futebol robótico da modalidade RoboCup Middle Size League da Universidade de Aveiro. Foi desenvolvida pelo grupo ATRI, pertencente à unidade de investigação IEETA. O presente trabalho baseia-se em trabalho desenvolvido anteriormente, tentando melhorar o desempenho da equipa. Cada robô da equipa CAMBADA é um agente independente e autónomo capaz de coordenar as suas acções com os colegas de equipa através da comunicação e da partilha de informação. O comportamento de cada robô deverá ser integrado na estratégia global da equipa, resultando assim em acções cooperativas de todos os robôs. Isto é conseguido através do uso de papeis(*roles*) e comportamentos(*behaviours*) que definem a atitude de cada robô e as acções que daí resultam.
Novos papeis foram desenvolvidos para complementar a estratégia de equipa, e alguns dos papeis existentes foram melhorados. Também foram efectuadas melhorias em alguns dos comportamentos existentes. É efectuada a descrição de cada um destes papeis e comportamentos, assim como as alterações efectuadas. O trabalho desenvolvido foi testado nas competições do Robótica 2008 (o desenvolvimento não estava ainda concluído) e por fim nas competições do RoboCup'2008. A participação da equipa no RoboCup'2008 é analisada e discutida. A equipa consagrou-se campeã mundial, vencendo a competição da Middle Size League do RoboCup'2008 em Suzhou, China.

keywords Multi-Robot Systems, RoboCup, Cooperative Behaviours, Robot Coordination.

abstract

Multi-robot coordination is one crucial aspect in robotic soccer. The way each team coordinates its individual robots into cooperative global actions define the foundation of its strategy.

CAMBADA is the RoboCup Middle Size League robotic soccer team of the University of Aveiro. It was created by the ATRI group, part of the IEETA research unit. This work is focused on coordination and strategy development for the CAMBADA team. It is built upon previous work and tries to improve the team performance further. In CAMBADA each robot is an independent agent, it coordinates its actions with its teammates through communication and information exchange. The resulting behaviour of the individual robot should be integrated into the global team strategy, thus resulting in cooperative actions by all the robots. This is done by the use of *roles* and *behaviours* that define each robot attitude in the field and resulting individual actions.

In this work, new roles were created to add to the team strategy and some of the previous existing roles were improved. Some of the existing behaviours were also improved to better fit the desired goals. Each role and behaviour is described as well as the changes made. The resulting work was put to test in the portuguese Robotica 2008 competition (while still in progress) and finally in the RoboCup'2008 world competitions. The performance of the team in the latter is analysed and discussed. The team achieved the 1st place in the RoboCup'2008 MSL world competitions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The RoboCup Initiative main goal is to promote the development of Artificial Intelligence, Robotics and related fields around the world. CAMBADA is the team of the University of Aveiro in the RoboCup Middle Size League (MSL).

The Middle Size League is played in a soccer field with $12m \times 18m$ by teams of 4 to 6 autonomous robots, each with a maximum diameter of 50 cm and a maximum height of 80 cm, weighing no more than 40 kgs.

CAMBADA has attended several national and international competitions in past years, faring reasonably well in the last year, achieving the 5th place in the RoboCup'2007 Middle Size League at Atlanta(USA) and 1st place in the national Robotica2007 competition. In this year the team won again the national competition Robotica2008 and conquered 1st place in RoboCup'2008 MSL championship, becoming world champion for the first time.

1.1 The problem

The soccer games proposed in the RoboCup MSL present great challenges to the robotic teams. To successfully play the game, the robots must coordinate and cooperate in their actions and correctly position themselves in the field (as a team) to prevent the opponent team from scoring. The robots must also exhibit accurate skills in dribbling and shooting the ball, so that they can successfully score goals.

Game strategies, positioning and team coordination play an important role in the Middle Size League soccer games. The development of stable, efficient strategies and accurate team coordination are crucial to achieve good results in the competitions.

1.2 Objectives

The objective of the proposed work is to improve the CAMBADA team performance and surpass previous achievements. To achieve this, it is necessary to improve the existing CAMBADA team strategies and team coordination behaviours by improving existing work in a way that can benefit the team performance. Existing work must be analysed, improved where needed and complemented by new developed work. The objectives are to improve the players at individual level and also in team cooperation and coordination. This is mainly done by improving existing *behaviours* and *roles* and also by the creation of new ones that will integrate the team strategy, resulting in a more efficient coordination and complete overall strategy.

1.3 Thesis structure

This work is divided into 8 additional chapters. Chapter 2 gives a brief description of RoboCup, its history and goals, and an overview of the several existing competitions. It also details the Middle Size League rules. In chapter 3 an overview of existing role coordination methodologies is detailed. Chapter 4 describes the CAMBADA team history and both software and hardware architecture. Chapter 5 details the modified behaviours. Chapter 6 describes the existing roles and the changes made. Chapter 7 describes the more coordinated natured roles of the team as well as some coordination aspects. In chapter 8, the team performance at RoboCup'2008 is analysed and results presented. Chapter 9 presents a small overview of the work done and final conclusions, future work is also discussed in this chapter.

Chapter 2

RoboCup: the robot world cup initiative

Figure 2.1: RoboCup

RoboCup is an international joint project to promote AI, robotics, and related fields. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined. RoboCup chose to use soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. The ultimate goal of the RoboCup project is by 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer [1].

The idea of soccer-playing robots appeared for the first time in 1993, originating a feasibility study that lasted two years. In April 1995, the study would culminate in the announcement of the introduction of the first international conferences and soccer games. The first official conference occurred two years later, in July of 1997 and was held in Nagoya, Japan. In the following years other cities throughout the globe housed the event: Paris, Stockholm, Melbourne, Seattle, Fukuoka, Padua, Lisbon, Osaka, Bremen and Atlanta. In 2008 the games

[1] `http://www.robocup.org`

were held at Suzhou, China.

The annual events had national and international media coverage and, over the years, attracted an increasing number of participants and spectators. In the past ten years the number of teams increased greatly from 38 in Nagoya (1997) to 321 in Atlanta (2007).

The RobCup World Championship and Conferences is the central pillar of RoboCup activities. It provides a world class event where researchers from all over the world can get together, evaluate research progress and participate in the competitions.

While soccer game is used as a standard problem, RoboCup expands beyond the competitions and has several more activities. Currently these consist of:

- Technical Conferences
- RoboCup International Competitions and Conferences
- RoboCup Challenge Programs
- Education Programs
- Infrastructure Development

The creation of the RoboCup Project focused on soccer, however RoboCup has increased its range of activities which are now divided into four main domains:

- RoboCupSoccer
- RoboCupRescue
- RoboCup@Home
- RoboCupJunior

2.1 RoboCupSoccer

The main focus of the RoboCup activities is soccer. Being a very popular sport worldwide it easily attracts people to the event, but it also brings a significant set of interesting challenges for researchers:

- A collective game, in which several agents/robots will interact through cooperation and competition

- Individualistic behaviour aspects, since each agent/robot must be able to identify relevant objects, self-localise, dribble, ...

- Cooperative elements, which will enable the existence of passes, complementary roles and all teamwork alike elements

- Dynamic and adversarial environment that permanently changes in real time, i.e. moving ball, moving teammates and opponents

The games are important and challenging opportunities for researchers to exchange technical information and assess technical progress. They also serve as a great opportunity to educate and entertain the public.

RoboCupSoccer is currently divided into the following leagues:

- Simulation league

  Autonomous software agents play soccer on a virtual field inside a computer. Matches have 5-minute halves and are played by two teams of eleven virtual agents. The game is based on a computer simulator that provides a realistic simulation of soccer robot sensors and actions. Each agent communicates with the simulation server sending communication and motion commands regarding the player it represents, and also receiving back information about its state, including the (noisy and partial) sensor observations of the surrounding environment. This is one of the oldest leagues in RoboCupSoccer.

- Small size robot league

  Disputed by robots smaller than 18 cm in diameter in a field with $5m \times 3.5m$. Matches have 10-minute halves and are played by teams of 5 robots with no on-board sensors. The field has an overhead camera which provides feedback to an external computer. Relevant objects (e.g., ball, own and opponent player locations) are distinguished by colour and coloured coded markers on the top of the robots. The external computer processes games information and sends the resulting commands to the team robots by wireless communications. This league focuses on the issues of multi-agent cooperation with a hybrid centralised/distributed system.

- Middle size robot league

  Middle-sized robots are limited to 50 cm diameter and a maximum height of 80 cm. The field size is $18m \times 12m$, in which teams between 4 and 6 robots play with an orange soccer ball. Matches are divided in 15-minute halves. Each match is controlled by a referee who has full authority to enforce the laws of the game. An assistant referee operates an application known as the *referee box* to send the competition signals such as start, stop, kickoff, freekick and so on to playing teams according to the main referee decisions. All sensors (mostly cameras) are on-board. Robots can use wireless networking to communicate. The robots are totally distributed and autonomous. Robots need to recognise objects and localise themselves using sensor information, decide which action to take, and control the motors and actuators autonomously. No external intervention by humans is allowed, except to insert or remove robots in/from the field.

- Standard Platform league

  The Standard Platform League extends the existing Four-Legged League (which used to be based on Sony AIBO robots). In this league all teams use identical robots. Using the same platform between teams enables them to concentrate on software development only, while still using state-of-the-art robots. The robots operate fully autonomously as there is no external control, neither by humans nor by computers. This league currently has two sub leagues:

  - 2 legged - uses the Aldebaran Nao humanoid robot
  - 4 legged - uses the Sony AIBO robot

- Humanoid league

  In the Humanoid League, autonomous robots with a human-like body, and human-like senses play soccer against each other. The robots are divided into two size classes:

  - KidSize - 30-60cm height
  - TeenSize - 80-130cm height

  The teams compete through several challenges in which they play against each other. These include:

  - Soccer games
  - Penalty kick competitions
  - Technical challenges

  Agent development is sought in several fields, as they have to deal with a multitude of challenging issues. An humanoid agent should be able to:

  - perform actions like dynamic walking, running, and kicking the ball (all while maintaining balance);

  - have visual perception of the ball, other players, and the field;
  - perform self-localisation, and act in coordination with teammates;

  These represent many of the research issues investigated in the Humanoid League. This may be considered the flagship league within RoboCup, as it is the competition that currently comes closer to the RoboCup final goal.

2.2 RoboCupRescue

The goal of RoboCupRescue initiative is to foster the science and engineering of search and rescue for large scale disasters, so that resulting research can directly contribute to society, and actually save people's lives.

The RoboCupRescue project promotes research and development in a socially significant domain at various levels including multi-agent team work coordination and physical robotic agents for search and rescue. It provides evaluation benchmarks for rescue strategies and robotic systems.

RoboCupRescue has common features with the game of soccer in various aspects, such as dynamic environment, incomplete and noisy information, but most importantly it also has features that are missing in soccer, such as logistics, heterogeneous agents, long-range planning, and emergent collaboration between different teams of agents.

The RoboCupRescue is currently composed of 2 leagues:

- RoboCupRescue Robot League

  Robots explore a disaster site, with mannequins with various signs of life. The robots, some under human control, must find and approach the victims, identify their signs of life and produce a map of the site showing where the victims are located. The aim is to provide the human rescuers with enough information to safely perform a rescue.

- RoboCupRescue Simulation League

  The league is composed of three competitions:

  - Virtual robot: simulated robots explore, map and clear a disaster area.
  - Agent competition: involves scoring competing agent coordination algorithms.
  - Infrastructure competition involves evaluating tools developed for simulating disaster management problems in general.

2.3 RoboCup@Home

RoboCup@Home focuses on real-world applications and human-machine interaction with autonomous robots. The aim is to promote the development of robots that will aid humans in common tasks. The scenario involves the home itself. Participants are given an environment that involves a kitchen and a living room. Contestants then demonstrate their robots' abilities in this environment. The basic home environment provided is the starting point. In the following years this will be expanded to include other areas of daily life, such as garden area, a shop, a street and other public places. The goal of RoboCup@home is to foster the development of useful robotic applications that are capable of successfully assisting humans in their everyday lives.

2.4 RoboCupJunior

RoboCupJunior is an educational initiative for young students. It is designed to introduce RoboCup to primary and secondary school children, as well as undergraduates who do not have the resources to get involved in the senior leagues of RoboCup.

RoboCupJunior offers several challenges, each emphasizing cooperative, problem-solving and task-achievement aspects.

- Soccer Challenge: teams of 2 autonomous robots (with a maximum of 22cm diameter and 22cm height) play games in a dynamic environment, tracking a special light-emitting ball in an enclosed $122cm \times 183cm$ field.
- Rescue Challenge: robots try to identify victims quickly and accurately within re-created disaster scenarios.
- Dance Challenge: one or more robots together with music, dressed in costume and moving in harmony.

2.5 Pushing the state-of-the-art

The RoboCup Project brings many great challenges to AI and robotics research. More importantly, it presents these challenges in a way that everyone can understand. It brings general interest, from young and old alike, into the promoted research areas. The competitions are fun and interesting for the general public, but also present interesting challenges from the research view point, as it brings researchers from worldwide to compete and compare developed work. This actively promotes global collaboration between investigators all over the world. The competitions that draw the general public to watch them also bring good chances for new sponsors for the several research projects in the relevant areas, helping researchers to acquire funding for their work. RoboCup is a worldwide project with a long range goal that constantly pushes the State-Of-The-Art in Robotics and Artificial Intelligence in a continuous global research collaboration effort.

2.6 Middle size league rules

Since the focus of this work is in the Middle size league competitions, the current rules [1] will be briefly explained.

The Middle size robot league rules are based upon the official FIFA Laws. Changes were made to enable the game to be played by the Middle size robots. The most significant changes will be described here:

- **Field** - The field is green and marked with white lines; it is 18 meters long and 12 meters wide; the goals are 2 meters wide; the field markings are detailed in figure 2.2

- **Robot** - robots are limited to 50 cm diameter and 80 cm height, the maximum weight of the robot is 40kg. Sensors are on board and the robots may communicate by WLAN technology. Robots must play autonomously without human intervention;

- **Teams** - Teams are composed of a maximum of 6 robots, 1 of which is the goalkeeper which is the only robot that can enter the Goal Area.

- **Ball** - The official tournament ball used is an orange FIFA standard size 5 soccer ball.

- **Match** - Matches are divided into 15-minute halves. Each match is controlled by a referee and an assistant referee that operates the *referee box* to send the competition signals such as start, stop, kick-off, free kick and so on in accordance with the main referee decisions.

- **Free Kick** - Free kicks, throw-ins, corner kicks and goal kicks are treated as indirect free kicks, i.e. a goal cannot be scored directly without the ball touching another player first.

- **Position Restrictions** - Only one robot from each team is allowed at any time inside the penalty area. In the opponent penalty area there is a restriction of a maximum of 10s intervals.

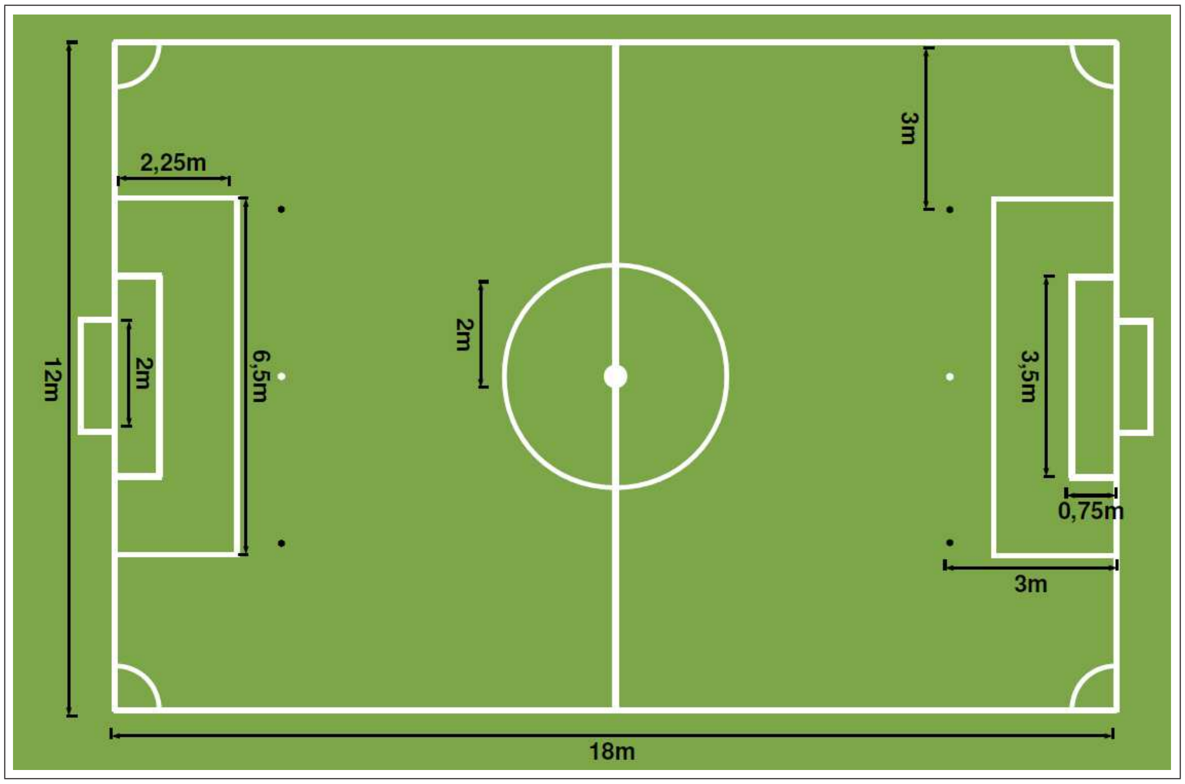


Figure 2.2: Official MSL field markings

Chapter 3

Coordination and decision methodologies

"Soccer is a complex game where a team usually has to meet several requirements at the same time. To ensure that in any game situation a team is prepared to defend its own goal, but also ready to attack the opponent goal, the various team players have to carry out different tasks and need to position themselves at appropriate strategic positions on the field." [2]

The soccer game proposed in the Middle Size League of RoboCup presents a wide range of great challenges for artificial intelligence and robotics:

- The development of autonomous intelligent agents to play the game
- Teamwork through multi-agent collaboration
- Strategy at individual and team level
- Surrounding world perception through sensor fusion
- Real-time reasoning to adapt to highly dynamic environments

An agent must observe its surrounding environments acquiring knowledge; it must adapt, react and also act upon the world, planning its actions towards successfully achieving its goals. Soccer is a collective challenge, thus each team is a multi-robot system. The individual agents must work together in cooperation towards the same goal of maximising the team performance. This usually involves strategic decision making and implicit or explicit communication mechanisms to exchange vital coordination knowledge and to share world information.

In the RoboCup Middle Size League, autonomous individual robots/agents perform collective actions, to face the existing cooperative challenges in the highly dynamic environment. Coordination and Cooperation are vital to achieve success [3, 4].

3.1 Roles

In robotic soccer there are several objectives that must be addressed at the same time. To achieve each of this objectives one or more tasks can be performed. A *role* exists with the aim of attaining one or more objectives by performing a set of actions that are available to it. For different types of objectives/tasks there can naturally be different types of roles. For example in rally sports where there are a pilot and a copilot, both perform different tasks (driving and reading directions), and yet they work as team. When a team has a multitude of goals to be achieved, it is natural to distribute the different actions required to complete each of the associated tasks by its members. Each element can carry out one or more roles inside the team to work towards an objective. This can easily be seen in soccer, as the several elements of the team carry out different roles: goalkeeper, defender, midfielder, striker among others.

Each of these roles defines the behaviour to be taken by the individual agent. By taking a role an agent will perform actions from the set of various actions available to him, with the aim of achieving the goals associated with its role. In the rally sports example the basic actions can be seen as steering, breaking and accelerating for the pilot and reading and talking for the copilot.

Roles allow for the definition of behavioural strategies to achieve one or more goals. This enables a team to divide its objectives into tasks, and in turn assigning these tasks to different roles that can be taken by its members.

3.2 Roles and coordination in MSL

Most prominent MSL teams have adopted some kind of *role*-based coordination, mainly due to the Stone and Veloso pioneering work [5] which inspired such methodologies.

Some approaches were made using static roles for each robot. However, since static role allocation lacks real adaptability to the high dynamic environments existing in the RoboCup soccer challenges, dynamic role exchange methodologies have been developed and are now utilised by most of the teams [6, 7, 8].

The dynamic role exchange adoption creates one challenge that sits at the core of each team strategy, the role assignment problem. The question of adapting the team behaviour to each situation in real gameplay, by efficiently assigning roles to each robot according to the environmental changes is similar to the task allocation problem for multi-robot systems that cooperatively work towards a goal, as shown by Gerkey and Matarić [4]. Gerkey and Matarić further discuss the role allocation problem in RoboCup and give an overview of existing team strategies in the several different leagues. According to Gerkey most of the teams that use dynamic role assignment, utilised a greedy strategy.

Uttori United [9] - The Uttori United team used a simple coordination mechanism: all robots pursue the ball, the first to grab the ball signals the teammates to not pursue the ball anymore.

ART team [7] - The ART team competed in the MSL league in 1999 and used an approach based on utility functions and explicit communications to coordinate the role allocation. Each robot would evaluate the utility of executing each role. This evaluation would be based on the local information about the environment of each robot. These values were then exchanged between robots. The available roles were ordered by importance, in such a manner that more important roles were assigned first. If there were fewer robots than roles, the less important roles didn't get assigned to any robot.

CS Freiburgh [2] - The CS Freiburgh team won the RoboCup MSL in 1998, 2000 and 2001. The role decision strategy is similar to the one used by the ART team. The players share the utility values for each role to maximise the sum of the utilities of each player. However, instead of exchanging roles immediately as done by the ART team, the player must check if there is another player pursuing the same role. It only takes the role if there isn't another player choosing the same role, or if the player with that role signals that it wants to change the role. As an addition to prevent oscillating situations, an hysteresis value is added to the player current role.

Cops [10] - The CoPS MSL team utilizes roles(Defender, Forward) and *subroles*(Left Defender, Right Forward, etc.) to coordinate actions. The roles are not exclusive to each robot. The robots share part of the information perceived from the world, this enables them to autonomously assign the roles and subroles. The list with the roles to be used is defined prior to the game and at half time. During the game, the robots are assigned dynamically to each role through a predefined role order, and according to a common algorithm that runs locally to each robot. Each role has subroles, these are assigned depending on the number of robots in each role. In addition there are special roles that take priority over the others. If a robot is assigned a special role (like Keeper, Attacker, Pass Player or Pass Receiver) it will still maintain its regular role as before, but will instead execute the behaviours of the special role. To achieve cooperative performance, a simplified version of Interaction nets is used to map behaviours to robotic agents. This mapping process is based on the knowledge represented in the world model. In order to cooperate, each robot gets assigned a different subrole.

Eigen [11] - The Eigen team coordinates actions through a permanent exchange of information between the robots. Each robot evaluates its own individual degree of objective achievement, called *self-evaluation*, which is then shared with the teammates. It then determines the *system-satisfaction*, which is the sum of the self-evaluation of the agents with higher values than itself. When a given system-satisfaction is higher than the desired level, the agent thinks that the intended objective is achieved and does not pursue that objective anymore. When the system-satisfaction does not reach the desired level, the agent thinks that the objective is not achieved and selects an action for achieving that objective. The Eigen MSL team was world champion in 2002, 2004 and 2005.

Chapter 4

CAMBADA

4.1 Background

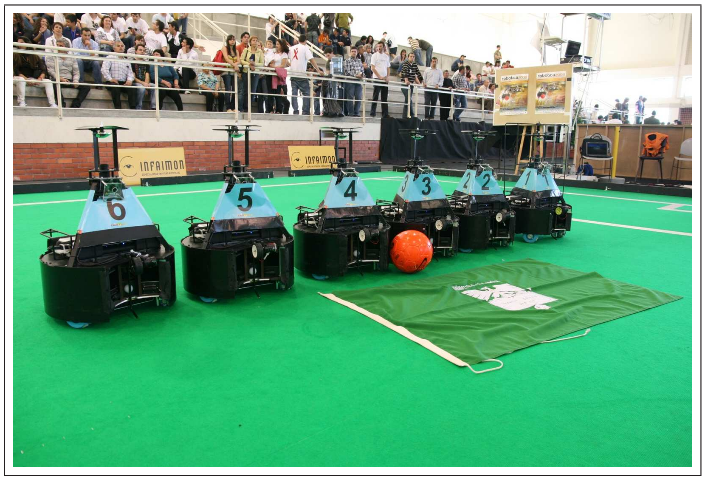


Figure 4.1: The CAMBADA robotic soccer team

CAMBADA is a RoboCup Middle size league soccer team developed by the University of Aveiro, Portugal. The project started officially in October 2003.

The CAMBADA project included the building of the mechanical structure of the robots and the hardware architecture. The development of the software addressed several areas such as image analysis and processing (obstacle, ball and field recognition), sensor filtering, information handling and artificial intelligence for multi-robot systems.

The team has attended several international competitions:

- RoboCup'2004 - hosted by Portugal in Lisbon from June 27 to July 5, 2004

- RoboCup'2006 - held in Bremen, Germany from 14 to 20 in June of 2006, where CAMBADA participated in a mixed team with ISocRob (IST, Lisbon)
- DutchOpen'2006 - at Eindhoven, Netherlands from 7 to 9 in April of 2006, where CAMBADA ranked in 7^{th} place, from a total of 12 teams
- RoboCup'2007 - at Atlanta, USA, from June 30 to July 10, 2007 where CAMBADA achieved 5^{th} place

The team also participated in several editions of the Portuguese Robotics Festival with improving results over the years:

- Robotica2004 - held at Porto, Portugal, from April 22 to April 25 in 2004
- Robotica2005 - held at Coimbra, Portugal, from April 29 to May 1, 2005
- Robotica2006 - held at Guimarães, Portugal, from April 28 to May 1, 2006 - 3^{rd} place
- Robotica2007 - held at Paderne, Portugal, from April 27 to April 30, 2007 - 1^{st} place

CAMBADA is the current holder of the National Champion title.

4.2 Hardware description

The CAMBADA robots were designed and completely built in-house. Each robot is built upon a circular aluminium chassis, which has around 485 mm diameter. This chassis supports three independent motors, each with a holonomic wheel, which allows omnidirectional motion. The chassis also supports an electromagnetic kicking device. The robot is powered by three NiMH batteries.

The physical composition of the robots can be divided into three layers:

- Low level sensing/actuating system

  This layer is placed upon the chassis base and is used to place all the electronic modules. It uses a set of microcontrollers interconnected by means of a network using FTT-CAN protocol[12]. The system is responsible for six main functions:

  - **Motion control** - using the three DC motors it provides holonomic motion functions to the high-level

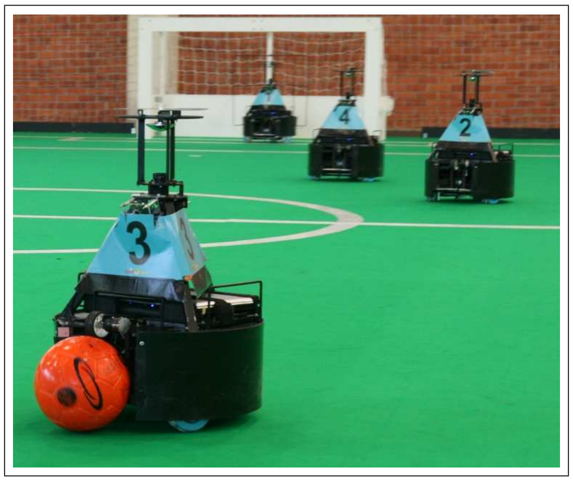


Figure 4.2: CAMBADA robots

- **Odometry** - combining the readings from the encoders in the three motors it provides robot movement information to the upper control level
- **Compass reading** - it provides the electronic compass readings for localisation purposes
- **Infra-red sensor** - the infra-red sensor detects if the ball is engaged by the robot and can be kicked.
- **Kicking** - it provides kicking and ball holding functions by making use of the ball sensor, the ball handler to dribble the ball, and of the electromagnetic kicker
- **System monitoring** - a control node that monitors the robot batteries and also the state of all the nodes in the low-level layer

With the current motion system, the robots can move at a maximum speed of 2.0 m/s. As mentioned, this is less than many of the other MSL teams, which can currently move at speeds typically between 2.5 and 4.0 m/s [13, 14, 15, 16]. All these functions are available to the high-level layer through a gateway node which acts as a translator between both layers, passing needed information only. The gateway synchronises the orders from the high-level with the low-level. In worst case scenarios this can take up to 70 ms.

- Main layer

  This second layer contains the main processing unit, currently a 12" notebook based on an Intel Core2Duo 2.0 GHz processor with 1024 MB of memory RAM. This unit is responsible for the image processing and analysis, sensor information fusion and real time reasoning and action. It is where the "brain" of the robot is.

The notebook is connected to the two vision cameras by firewire, from where it acquires visual information of its surroundings. It is also connected to the lower sensing/actuating system through an USB cable. This is used to receive odometry, compass and battery information and also to send the motion commands (as previously detailed). The notebook is connected through wireless communications to the referee box, to his teammates and also to a coach agent.

- Top layer

  At the top of the robot stands a hybrid vision system. This consists of a perspective frontal sub-system which is composed by a standard firewire camera pointing forwards, and an omnidirectional sub-system supporting a standard firewire camera, that relies on a catadioptric camera setup with an hyperbolic mirror. This layer also includes the physical electronic compass that was installed this year. The new rules in the MSL remove the different colours in the goals, which were previously used to identify the field side. This disambiguation is now performed using compass information. Note that the compass readings are done by the lower layer, but the compass was installed on top of the robot in order to minimise electromagnetic interferences from motors and kicker located at the lower layer.

4.3 Software architecture

The general architecture of the CAMBADA robots has been thoroughly described [17, 18, 19]. The software system in each player is distributed among the various computational modules depicted in figure 4.3. High-level functions run on the Linux based notebook. Low-level functions run partly on dedicated microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) has been adopted.

The software of CAMBADA players is composed of several different processes that exchange information through the RTDB. Each process is responsible for different tasks: image acquisition, image analysis, integration/decision and communication with the low-level modules. The order and schedule of activation of these processes is performed by a process manager library called Pman [20].

4.3.1 RTDB

The RTDB is a data structure where players store their world models and relevant coordination information. This information can be accessed by the several processes. The shared

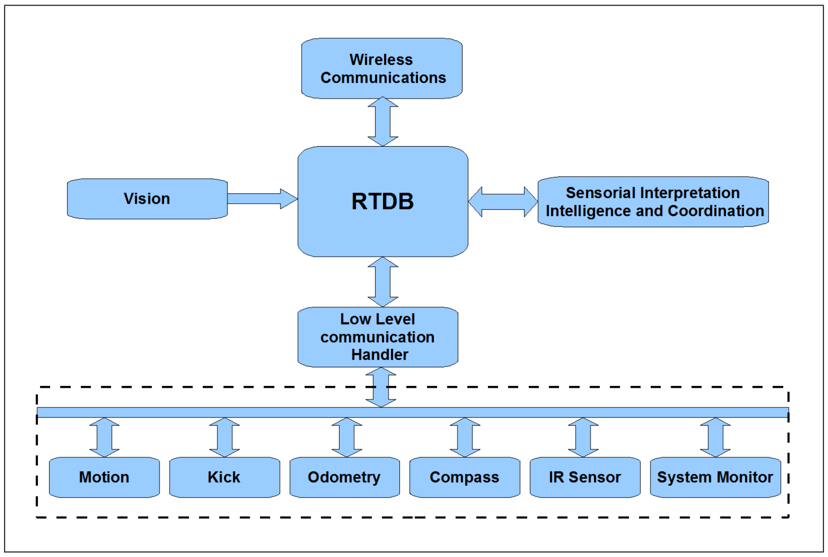

Wireless Communications
Vision
RTDB
Sensorial Interpretation Intelligence and Coordination
Low Level communication Handler
Motion
Kick
Odometry
Compass
IR Sensor
System Monitor

Figure 4.3: Software Architecture

part of the RTDB is updated and replicated in all players in real-time, with a periodicity of 100ms. This is guaranteed by an adaptive TDMA transmission protocol [21, 22] on top of IEEE 802.11b, developed to reduce the probability of packet collisions between teammates.

4.3.2 Vision

The vision process retrieves the cameras frames and does the subsequent processing. The module makes use of radial search lines to analyse colour and contrast information in the image. This helps to identify the relevant objects in the field and their positions (ball, white lines, obstacles). This information is then made available to the high-level layer by storing it in the RTDB. This vision system detects objects, and field lines on a radius of up to 5 m [23].

4.3.3 Low-Level communication

The hardware communication process gathers information from the hardware micro controllers (odometry, infra-red sensor and compass) and makes it available through the RTDB. It also relays the commands to the microcontrollers from the agent, so that they are executed through the actuators in the low-level. In worst case scenario the delay between the orders being issued and performed can be up to 70 ms.

4.3.4 Sensorial interpretation, intelligence and coordination

This is the main process in a CAMBADA player. It can be seen as the brain of the player. It gathers and integrates sensor information, decides which actions to take and which orders to issue to the actuators in the lower layer. This is where the individual skills and the coordination mechanisms of the players are implemented. The scope of this work is confined to this module which will be detailed in the following subsection.

4.4 CAMBADA player high-level

The high level functions of the CAMBADA player can be divided into three main modules:

- Sensor fusion

  Combines sensor information retrieved from the robot own sensors and also information received from other robots to create and update the world state database, an internal world representation.

- High-level decision and coordination

  This is the module responsible for the team behaviour and decisions, it analyses the current situation through the information in the world state and chooses the individual attitude to be taken by the agent in order to maximise the team performance. This is done by choosing the active role.

- Basic behaviours

  Provides the set of primitives that the active role uses to control the robot. The several behaviours represent the set of actions that can be taken by the robot. They are responsible for the orders to be issued to the low-level layer.

A player chooses its actions based on the knowledge it has. In the CAMBADA player case, this knowledge is stored in the *worldstate*. The worldstate is the player perception of the surrounding real world. It is here that all the information is maintained and made available in a structured manner.

The CAMBADA high-level functions are based on the concepts of role and behaviour [5]. Roles reflect given attitudes like being a striker or a defender and are responsible for selecting the active behaviour at each time step. The decision of a role is based in the worldstate current information and also on the role own internal state.

Behaviours represent the basic sensorimotor skills of the robot, like moving to a specific position or kicking the ball. The execution of each behaviour will result in orders being issued for the lower-level layer. These include individual velocity orders for the 3 axis considered in the robot movement (linear X and Y velocities and Angular velocity). The active behaviour can also order the robot to kick the ball with a given power. The movement controllers of the behaviours are an implementation of PID controllers [24]. Each behaviour has its own PID values (P,I and D) tuned for the desired performance.

Roles and Behaviours are the basic building blocks of the CAMBADA strategy.

4.4.1 Lifecycle

The lifecycle of the CAMBADA player can be divided into three steps: sensor fusion, decision making and actuating.

The information in the worldstate is updated every cycle before the player decides and actuates. In order for the information to be available for the high-level decision module it must first be filtered and processed.

The first step is to gather the raw information that was written in the RTDB by the several sensors. This information is then filtered, processed and integrated in the worldstate. Part of it is made available to the teammates by writing it in the shared area of the RTDB. This is followed by the integration of local information with the data obtained from the teammates.

The information present in the worldstate includes player position, ball position, ball velocity, teammates positions and obstacles positions among others.

The next step is the decision process. In this step individual and cooperative attitudes and actions are determined. The player decides which actions to take based on the information provided by the worldstate.

The decisions of the player result from analysing the game situation, the teammates positions and their chosen roles. The player will choose one role according to that information, and this role will define an attitude that will be carried out by the player. The execution of the role will determine which behaviour will be used.

The third step is actuating. The execution of the chosen behaviour will result in orders being issued for the lower-level layer, these include target movement speeds and possibly a kick command.

Chapter 5

Behaviours

As stated in the previous chapter, behaviours are the sensorimotor action primitives that the active role uses to control the robot, like moving to a specific position or kicking the ball. The execution of the behaviour will result in commands being issued for the lower-level layer. These include individual velocity commands for the 3 components considered in the robot movement (linear X and Y velocities and Angular velocity). It can also command the robot to kick the ball with a given power value (this value can range from 10 to 50).

By the time this work started the CAMBADA team had several behaviours:

- *Move* - allows the robot to move to a field position in relative coordinates;
- *MoveToAbs* - analogous to the previous but using an absolute field position;
- *Dribble* - allows the robot to move with the ball engaged
- *Kick* - responsible for aiming and kicking the ball;
- *Goalie* - the basic behaviour of the goalkeeper;
- *Stop* - stops the robot.

During the timeframe of this work, three new behaviours were developed by João Silva as detailed in [24]. João's work also includes improvements to some of CAMBADA software key areas like sensor fusion and integration. The behaviours added by João were:

- *PassiveInterception* - makes the robot move into the predicted ball path without directly approaching the ball.

- *ActiveInterception* - calculates an optimal interception point in the ball predicted path and moves to it in a way that the robot can grab the ball.
- *CatchBall* - this is the behaviour used to receive a pass, similar to the `PassiveInterception` behaviour, but also moves back when catching the ball to reduce the collision impact.

To successfully improve the CAMBADA roles, it became necessary to modify/improve some of the behaviours. This is a result of the intrinsic dependency between roles and behaviours performance.

The behaviours are the basic actions available to the roles. If these actions are not optimal or do not translate the correct intentions of the role, then the resulting performance will be poor. Roles and behaviours must work well together.

In this chapter improvements made to some of the existing behaviours will be detailed.

5.1 Move

To perform the robot movement there are two behaviours, the *Move* behaviour and the *MoveToAbs* behaviour.

The CAMBADA robots have holonomic motion as previously explained. This has great advantage allowing the robot to move in any direction with any orientation. When ordering the robot to move to a destination position, an additional position is given, which indicates to where the robot should be facing while moving.

The *Move* and *MoveToAbs* behaviours are used by the roles to order the robot to move to an intended target. This target can be a position on the field or a given known object. The current known objects are: the ball, the opponent goal, the own goal and the field centre. It also accepts the target to which the robot should be facing; this second parameter can also be a position or a known target. Optionally a maximum velocity can be defined.

5.1.1 Rotational concerns

While using these behaviours it was noticed that when the robot tried to change its orientation to face a given point while moving (holonomic motion), the resulting movement was, not a straight line, but a curved one [24], taking a path longer than needed and consequently taking more time to reach its destination.

These trajectory errors are probably caused by the sensing-actuating delays, that cause the holonomic motion module to perform its calculations based on outdated visual information and localisation.

The delays are due to the delays of the several processes involved:

- the inherent image acquisition time, also experienced by other teams with similar robot camera configurations (around 150 ms);
- the time it takes to do the image analysis and the time it takes for the agent itself to run and determine the robot response, roughly 30 ms overall.
- the time it takes to synchronise the agent orders with the low level motors, which can take up to 70 ms in worst case scenarios.

This results in a roughly 250 ms delay between sensing and actuating.

It was noted that for small orientation adjustments (less than 45 degrees) the robot did not suffer the unintended curved trajectory. So in order to prevent the curved trajectory from happening, a slightly different approach was devised for these behaviours. When having to perform large adjustments to the robot orientation, it would be better to first perform a large adjustment without translation speeds, and only then to move to the intended target while performing minor corrections to the orientation as needed.

To achieve this, the robot angular speed response was augmented (by increasing the P value of the PID filter for the rotational movement from 1.0 to 1.6) for these behaviours, as it was intended for the large orientation adjustments to be performed as fast as possible.

This increase in the rotational response was coupled with an additional method to better control the robot rotation movement. After the PID filtered values are calculated, the resulting orders are then filtered by a very specific code to prevent high speed movement while the robot is rotating at high angular velocities:

```
If( |velA| > Π )
    {velX = 0.0; velY = 0.0;}
Else if ( |velA| > Π/2.0 )
    {velX = velX * 0.5; velY = velY * 0.5;}
Else
    {velX = velX; velY = velY;}
```

This causes the robot to first perform a very fast rotation to the intended target while not performing any translational movement (if the angular velocity is greater than π rad/s) or only moving at half speed (if the angular speed is greater than $\frac{\pi}{2}$ rad/s). Only when minor rotational adjustments are needed (angular speed smaller than $\frac{\pi}{2}$ rad/s) it will move at the speed calculated through the PID filter.

5.1.2 Obstacle and Ball Avoidance

Obstacle and ball avoidance are the methods used by the move behaviours, while moving, to avoid contact, with either the ball or with field objects (other robots) or both.

Obstacle avoidance is an important aspect of these behaviours. It is used by default in every move order, and usually only deactivated when trying to catch the ball (as avoiding opponents wouldn't be very wise). Obstacle avoidance is needed to prevent strong collisions, which can make the robots malfunction or crash and would hinder the team overall reliability.

Ball avoidance can also be enabled. This is only used when the game is stopped and the robots can't touch the ball until the game comes into play [1].

The vision software module creates a list of points with the positions of all the obstacles it can detect. This list is then made available for the high level decision layer. These obstacles are used to check if the robot surrounding space is free.

The intended target position to which the robot should move is passed on to the move behaviours. If obstacle avoidance is activated the field is scanned to determine an obstacle-free direction as close to the target direction as possible. To perform this analysis a given number of equally separated directions is considered. These directions divide the robot surrounding space in slices, one slice for each direction considered. If there is an obstacle closer to the robot than the predefined distance in a given slice, the corresponding direction is marked as being obstructed.

After all the obstacles have been checked across the several slices, the first free direction, closest to the target direction, is chosen and the direction in which to move is determined.

To perform ball avoidance the same strategy is used. A small set of obstacles is added on the obstacle list. These obstacles are created in a circular fashion surrounding the ball position and with a distance equal to the ball's radius from the ball. These obstacles are then used together with the remaining obstacles as previously explained.

Currently for this method 8 directions are used and obstacles until a distance of 1.5 meters are considered.

This obstacle avoidance strategy is very simple in concept and does not perform any kind

[1]See Section 2.6 - MSL rules

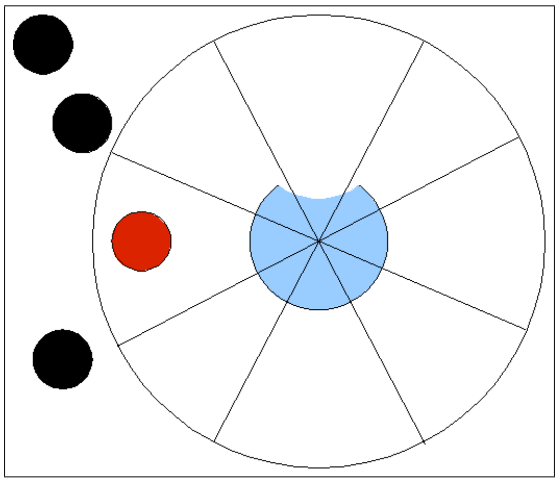


Figure 5.1: Obstacles analysis by direction slices - an obstacle in a slice will inhibit the robot from choosing that direction

of collision-free path planning as an A* algorithm would do. Then why was it chosen instead of a different strategy that could plan the path all the way from the robot position to the target position?

There are some reasons that lead to the choice of the simpler strategy:

- The obstacle detection algorithm used in the vision module only detects obstacles in the vicinity of the robot (at a maximum of 5 meters), and the technique that is utilised to achieve this only detects the closest obstacle per direction. This kind of information isn't enough to perform an accurate path planning, as there exist in the field many more obstacles than those the robot can immediately see, and the range at which we can detect them only allows for small distance planning.

- There is also the fact that most obstacles are fast moving robots, so most of the path planning is not very useful and can easily change from cycle to cycle, possibly producing intermittent and incoherent resulting targets which could possibly cause the robot to act in a confuse manner.

- It must also be taken into consideration that the simpler strategy is faster to execute while the path planning strategy could take significantly more time.

These reasons lead to the implementation of the more reactive approach that showed good results and met the intended objectives: create a reaction mechanism that would avoid collisions when possible or at least reduce the strength with which these occur, while trying, whenever possible, to move closer to the target.

5.1.3 Move and MoveToAbs

When the separation between *Move* and *MoveToAbs* was created the intention was to simply divide parameter types (the *Move* behaviour accepts relative targets or known objects, both for the target position and looking position, while the *MoveToAbs* accepts absolute positions or known objects). However, now the difference between both behaviours has spanned larger than this as they are used with different purposes.

The standard *Move* is used when approaching the ball, which requires special care for not going too fast as the robot can easily push the ball away instead of grabbing it, and also takes into account situations when the robot is going after the ball at roughly the same speed the ball is rolling away, which usually would result in the ball going outside the field or being caught by an opponent faster robot. This is achieved by fine tuning the PID filters so that the robot performs in a desirable fashion, taking into account the referenced situations.

After several tests, the adequate PID values found for the translation movement were P = 1.4 to allow the robot to approach the ball smoothly. A minimum output of 1.0m/s was specified, so that it keeps chasing the ball at a reasonable speed when it is close to it, as the ball can be moving away. I and D values were set to 0 in the translation movement. For the rotational movement the P value was 1.6, I value of 0 and D value of 2.2.

The *MoveToAbs* behaviour doesn't take these concerns into account. It is designed to move to an intended destination as fast as possible, or as fast as requested. It is not an adequate behaviour to perform the ball approximation. But it is more responsive and aggressive than the standard *Move*. The only concern in calibrating the PID filter for this behaviour was to prevent it from overshooting the final destination target, as that would take more time than what would otherwise be needed. The PID values found for this behaviour translation movement were, P = 2.0 for a fast response, I = 0 and D = 0, and for the rotational movement P = 1.8, I=0 and D = 0.

5.2 Dribble based on utility field

The *Dribble* behaviour is used to progress in the field while keeping the ball engaged. It does so along a direction passed as a parameter. This parameter value usually was the opponent goal direction. This resulted in a simple and not very intelligent behaviour. When the robot got hold of the ball, it would simply go in a straight line towards the opponent goal, performing some obstacle avoidance if needed.

In the MSL games, different teams have different strategies and each team layout on the field is different. It is easily perceived that most teams concentrate themselves in the centre of the field and around the penalty marker, usually leaving the sidelines and adjacent area free. In order to take advantage of this situation, a different approach was created to determine the best direction to follow with the ball. This new method is based on utility values across the field, allowing some degree of customisation and resulting in a more intelligent final behaviour and trajectory.

5.2.1 How the utility field is defined

A file with a small table with an arbitrary number of rows and columns representing the field is created. In this table, rows represent the field length and columns represent the field width. The table is filled, assigning values to each position in the field. This table is read by the agent at start up time and the information is scaled to a full size field that will be used by the algorithm, this is done by expanding the table size to the actual field size, resulting in a table with one cell per square meter. In this expansion the existing values are replicated in the initial proportion to meet the final size. In order for the method to work properly, the values are smoothed through several passes of a mobile average and at the end the values are normalised (to a 0.0 to 1.0 scale). This creates soft transitions across the entire field, preventing the existence of plane valued areas, which could cause the robot to not move.

While progressing through the field the robot will always try to go in the direction of a higher valued surrounding position until eventually the active role tries to kick.

To determine the direction that will be passed on to the *Dribble* behaviour a worldstate function is executed. This function uses a similar approach to the obstacle avoidance method described earlier in the move behaviour description. It uses a given number of directions with a predetermined length that divide the robot surrounding in slices, one for each direction. The value of the utility field at the edge of each slice is assigned to the corresponding direction. Then all the slices are checked for obstacles. If there is an obstacle that is within one slice, the corresponding utility value of that direction is decreased by 1.0. After these steps have been executed, the direction with the highest utility value is chosen. As the utility values are never higher than 1.0, this results in free directions being chosen over obstructed directions, no matter what the initial utility value was.

The full sized field has a resolution of a value per square meter. Although this seems a low resolution it is enough to provide the directions needed. The method utilised to determine the highest valued surrounding position, based on sensors, may seem inaccurate. However coupled with the low resolution field that has been smoothed it will perform as expected and

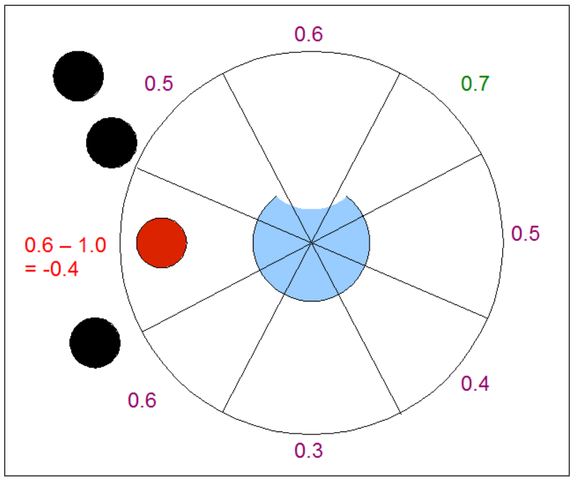


Figure 5.2: Example of how the utility field is used; the red obstacle reduces the utility of the slice it is in; the direction selected is the one with the highest value (in green).

avoid obstacles as efficiently as the move behaviour. The robot localisation system easily produces errors up to 20 cm of magnitude. Creating a more detailed, higher resolution utility field would prove of no advantage and would in any other case be above the scope of what is intended with this function: to provide a direction in which it is reasonably safe to progress with the ball while trying to reach a higher valued position in order to shoot in goal. The values used to compose the field define through where the robot should preferentially try to advance. This allows for some degree of customisation and adaptability to each opposing team field layout.

For this method 16 different directions are considered, each with a 4.0 meters length.

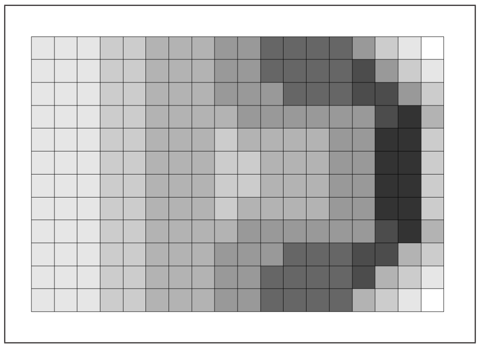


Figure 5.3: An example of an utility field, darker areas have higher values.(Attacking to the right side)

5.3 Kick

This behaviour is used when trying to shoot or pass the ball. It will kick the ball to an intended point or target if specified, else it will choose the best point in the opponent goal to maximise goal chances. It is the behaviour responsible for the aiming and kicking.

There are 3 fundamental parts to the *Kick* behaviour: target choosing, aiming, and kick power calculation. How the behaviour achieves each of these crucial aspects will be detailed below.

5.3.1 Target choosing

In the *Kick* behaviour the target can be passed as an argument (in two different ways), or if no target is passed the behaviour will choose an appropriate target in the opponent goal.

- Target Point

  The target is passed as a position in the field. Optionally it can have a flag indicating it is a pass to a teammate which will cause the *Kick* behaviour to use lower power values.

- Target Goal

  The target is the opponent goal. A parameter chooses between centre, left or right side of the goal.

- No Target

  If no target is selected, the *Kick* will automatically select a target. If it is in the left side of the field it will chose a predetermined target on the right side of the goal and vice versa. The target point selected is in the opposite side because the goalkeepers usually position themselves near the post that is closest to the ball, thus targeting a point near the other post allows a good chance of shooting a volley over the goalkeeper and into the goal.

5.3.2 Aiming

For the robot shot to be accurate and reach its destination, we need to consider 3 aspects: if the robot is aligned with its intended target; if the robot is not rotating; and if there are no obstacles in the ball path.

Alignment

The robot target alignment is obtained with robot movement in a circular fashion to match the desired orientation towards the target.

The robot aligns itself with the intended target by performing rotational and sideways movement simultaneously, resulting in a circular trajectory. The rotational movement unaccompanied by the sideways movement could cause the ball to roll away, as with this circular movement the ball is kept engaged in the robot.

The robot also moves forward at a small velocity, as it helps to keep the ball engaged and gain terrain while aiming.

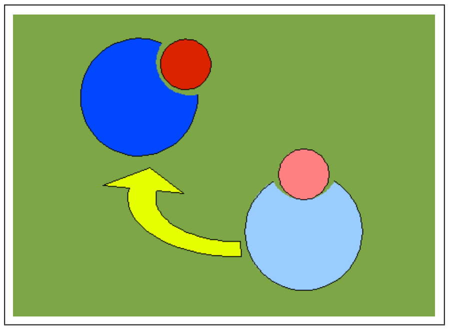


Figure 5.4: Example of the circular movement performed by the *Kick* behaviour in order to avoid losing control of the ball

The difference between the current and the target orientation is passed to the PID controllers of the *Kick* behaviour, that will make for the necessary response in the robot accelerations so that it moves into a position with the right heading by moving as described before. Aligning the robot with the target is a precision task. To achieve the required smooth (yet fast) movement, the PID type filter is used (as is in all behaviours) and P, I and D values were determined for this behaviour. Upon several tests, it was found that a P value of 2.1 and D value of 0.0 with an I value of 0.0 produced the best results and provided a good balance between smooth and fast movement to achieve the desired alignment.

Alignment check

The alignment with the target is always verified before shooting. Currently a maximum difference of 5 degrees is allowed between the direction in which the target position is and the robot actual facing direction. It was found that using a smaller value would not improve aiming as the kick can easily deviate within that order of magnitude. Tightening the alignment slack could cause the robot to intermittently fail the alignment check due to localisation noise, which could result in the robot not kicking or delaying the kick unnecessarily.

When shooting in goal, before allowing the kick, the behaviour executes a goal line interception verification function: *shootInGoal* - This function considers the opponent goal line segment, and checks if the current predicted ball trajectory intersects this line segment. This is made by extending a line from the robot position to the direction in which the robot is facing. If there is no intersection between the lines the kick is inhibited, but if the lines intersect the shot is allowed.

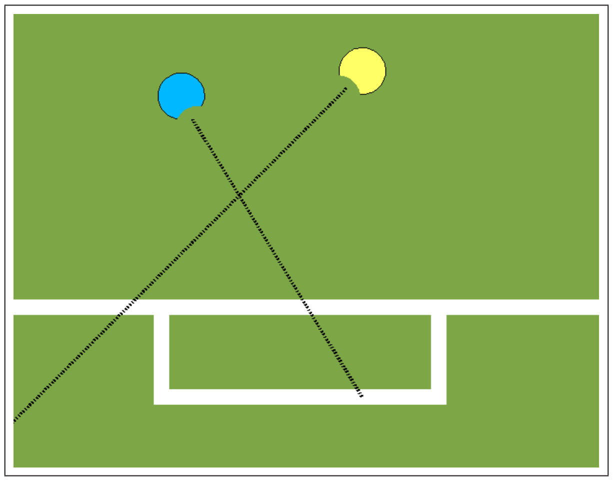


Figure 5.5: ShootInGoal verification example, the blue robot returns a positive check, while the yellow robot returns a negative check

Rotation

If the robot kicks the ball while still rotating at some considerable speed, then all the target alignment work will be meaningless. For a good shot to take place, it is required that at the moment the ball is kicked, the robot isn't rotating.

Rotation is obviously necessary to perform the aiming duty, so in order to be sure the robot is not rotating while shooting, it is necessary to watch the robot angular velocity, and inhibit any kick from taking place if this value is not near 0.

If the robot is rotating, its orientation value will vary accordingly. However due to localisation problems, the robot can be stopped and still variations on the orientation value would occur.

The ball kick is inhibited unless the robot orientation angle has remained stable for some small amount of time. Variations from cycle to cycle of the robot orientation value are measured. If these variations exceed a given threshold, then the *notRotating* time counter value is reset. Otherwise the counter is increased by the amount of time it passed between current and previous cycle executions. Only when this counter exceeds a certain threshold can the rotation inhibition be disabled, and so the robot can shoot. Should the angle vary more than what is deemed acceptable, the counter is reset, and so the kick inhibition is back in place.

After some series of tests the best value found for the maximum angle variation threshold was 1.5 degrees per cycle (which results in a maximum angular speed of $\frac{\pi}{2}$ rad/s) and for the minimum time without angle variation threshold was 200ms. These values were found to be a good balance between swiftness and stability, as the robot didn't wait too long after rotating nor did it shoot while still rotating.

5.3.3 Power selection

The robot kicker mechanism accepts integer values, ranging from 10 to 50, for the kick power. In order to maximise our chances of goal the ball should reach the opponent goal with a height close to 80 cm (the robots maximum height), as this would result in the ball going over the goalkeeper and would be very hard to defend. Although the rules allow the goalkeeper to extend its body beyond the 80 cm limit for short periods of time, very few goalkeepers have this feature (currently only the TechUnited and ISEPorto teams have such a goalkeeper [25], and it tends to be very hard to use efficiently).

When determining the right power for each distance, one must notice that the same power value results in different kick strength for each robot, so whatever the technique used to determine the right amount of power for each distance, it must cope with robot variations.

To help the *Kick* behaviour choose the right power value to kick the ball to a given distance and to the preferred height, a function was created. Given the target distance, this function determines the necessary power so that the ball reaches the target with a desired height close to 80cm. This function also takes possible bounces into consideration as the target can be farther than what the kick can reach with a direct, bounce free shot.

To determine the correct power for each distance in each robot, a set of polynomial approximations were created and then calibrated for each robot.

The situations that were taken into consideration were shot with no bounce and shot with one bounce. Different polynomial functions were created for each situation. Kicks with 2 or more bounces were not taken into consideration as they wouldn't be able to reach the desired height. So, beyond a certain distance, the robot just kicks with maximum power so that the ball reaches its target as fast as possible.

To determine polynomial function coefficients, each robot is tested individually and the values of the kick power are determined at some chosen distances. Values are taken, starting from 2 meters with one meter interval up to a distance of 9 meters. At this distance the necessary power usually reaches (or surpasses) the maximum value possible.

After the values are determined for these chosen distances the polynomial functions are determined (one for direct shot, one for shots with one bounce). Usually the direct shot is possible at as far as 7 meters, and the one bounce shot usually ranges from 7 to 9 meters. For each function the determined kick powers are used to calculate the polynomial coefficients that are then used to calculate the final power value. For distances smaller than 2 meters, the power value returned by the polynomial function for 2 meters is used, the only concern at this point is for the ball not to go over the goal.

These kick power calibrations are made offline and the values are then loaded into the robots. At the moment of the kick, the target distance is utilised together with the polynomial functions to calculate the necessary power to successfully shoot the ball into the target.

Since the robot calibrations are made with the robot stopped, a small adjustment is made to the power value when the robot shoots while moving at speeds greater than 0.3 m/s. This is made by sending a distance 0.35 smaller than what the target really is. This value was found to compensate the power value adequately.

5.4 Goalie

The goalkeeper is a very important part of an MSL team. In the MSL league most robots kick the ball in a volley shot manner which, given the hyperbolic omnidirectional camera setup [2], makes it harder to accurately determine the position of the ball when airborne. Also the ball shots tend to be reasonably fast, even more with ground shots [25]. This is added to the fact of having to cope with the known sensor-actuating delays (as previously detailed). A MSL goalkeeper strong point should be the defencive positioning, as quick reaction after the shot is taken is not possible for most of the kicks.

5.4.1 Positioning

The goalkeeper positions itself along the goal keeper line: a line segment placed 35 cm ahead of its goal, which spans the same length as the distance between the goal posts minus

[2]See Section 4.2 - Hardware description

the robot radius (take into account that the goalkeeper robot radius is greater than the others robots due to the goalkeeper frame). The 35 cm are the minimum distance found that can cope with some localisation error and still allow for the robot to rotate without colliding with the goal. The importance of staying close to the goal line is because it creates a better chance at defending the volley shots that otherwise would go over the goalkeeper.

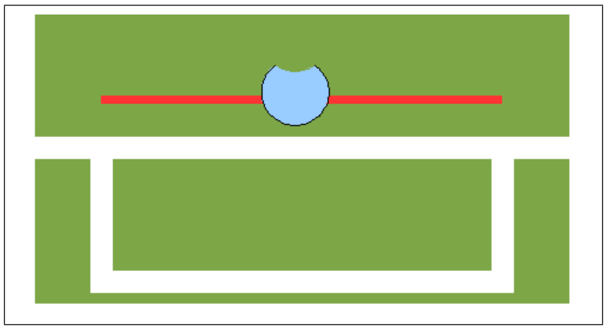


Figure 5.6: Goalkeeper positioning line (in red)

Concerning the positional problem there are four situations that were considered:

- The ball is within close proximity

  If the ball is within the 0.5 meters ahead of the robot, it will move to the position in the goal keeper line that is perpendicular to the ball.

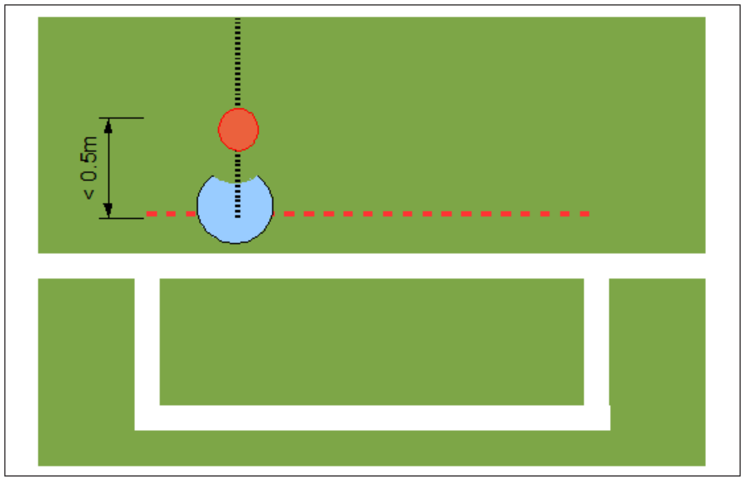


Figure 5.7: Goalkeeper positioning when the ball is at less than 0.5m (vertically) from its positioning

- The ball trajectory doesn't currently intersect the goal line

  If the ball trajectory isn't intersecting the goal line, the positioning is calculated through an attraction based positioning. The ball attraction value is multiplied by the ball x-axis positioning, determining the goalkeeper own x-axis positioning along the goal keeper line. This attraction value varies according to the ball distance to the goalkeeper:

  - Farther than 6 meters, attraction value of 0.10. This causes the robot to stay mostly on the goal centre when the ball if far away. Should a long shot occur and the ball, being airborne, becomes not visible, the robot is in the best guessed position to defend it.
  - Farther than 3 meters, attraction value of 0.25. Analogous to the previous while being fairly more responsive to ball movements.
  - Closer than 3 meters, attraction value of 0.65. This high value results in the goalkeeper making a good goal line cover following the ball movements closely and aggressively, diminishing open angles to a reasonable minimum.

- The ball is moving and it will intersect the goal line

  If the ball trajectory intersects with the goal line, then the goalkeeper moves to the position in the goal keeper line in which the ball trajectory is intersected. This places the robot in the most probable place to better defend the ball.

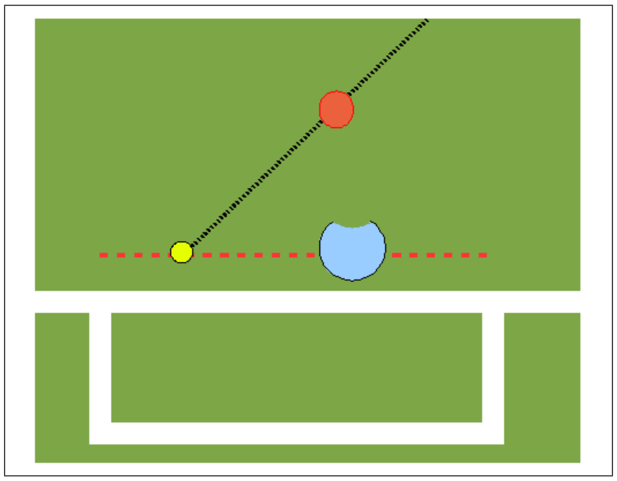


Figure 5.8: The goalkeeper determines the ball trajectory interception point (yellow circle) and moves towards it

- The ball is between the goal keeper and the goal line

  This is a very complex situation as the goalkeeper is already very close to the goal line. Moving the robot can easily cause the ball to go into the goal, especially since trying to retrieve the ball is a complicated process due to the goalkeeper frame which is hard to account for, when performing movement and rotations.

Moving the robot across the goal line severely diminishes the localisation algorithm accuracy, and potentially leads to the robot becoming lost, which in this situation could result in inadvertently scoring in the own goal. Therefore, in this case, the robot stops. Since the ball is in the goal area, no other robot can legally touch it or touch the goalkeeper, and after a short period of time this situation becomes a drop ball foul, which in spite of not being a good strategy is better than facing the odds of scoring int the own goal.

5.4.2 Bouncing ball

When the ball is kicked by a robot and it starts bouncing, it becomes very difficult for the vision system to accurately determine the ball position. This causes the ball known position to change rapidly. This happens because the robot vision can determine the current ball direction, but not the correct distance at which the ball is. This is due to the current technique used to determine the ball distance relying on the ball height in the image. Since most of these kicks are made at our goal, this error inducing bouncing effect may cause severe problems to the *Goalie* behaviour, since the behaviour relies heavily on the ball position and speed vector. With these errors causing erroneous information the *Goalie* will consequentially behave in an undesired manner, which can greatly affect its effectiveness.

In order to counter this problem two strategies were developed:

- Ball triangulation with a teammate. Since ball direction is accurately detected even if the ball is bouncing and all the robots share their ball positions (albeit with a small delay for communication of 100 ms [3]) it is possible to use the *Goalie* and another teammate information to accurately triangulate the ball position. The direction in which the ball is, is given by the *Goalie* relative ball position. To determine the right distance a teammate is chosen to give the additional required information.

  A search is made among all the teammates to find the robot that is closer of being perpendicular to the line made by the *Goalie* and the ball velocity direction. Then using this teammate relative ball position a second line is created, and the two lines intersection point is the new estimated ball position. This new ball position does have its own error, since the teammate information can be, in worst case scenario, roughly 400 ms old, which if the ball is moving fast can generate a considerable large error, up to around 2 meters.

[3]See section 4.3.1

However, having this educated guess at the real ball position is better than having the previous hazardous positions that were returned due to the ball bouncing.

In case there are no other teammates (not very likely) or that all the active teammates are in an almost collinear position (the angle formed between the two lines is smaller than 15 degrees) the ball triangulation is not utilised.

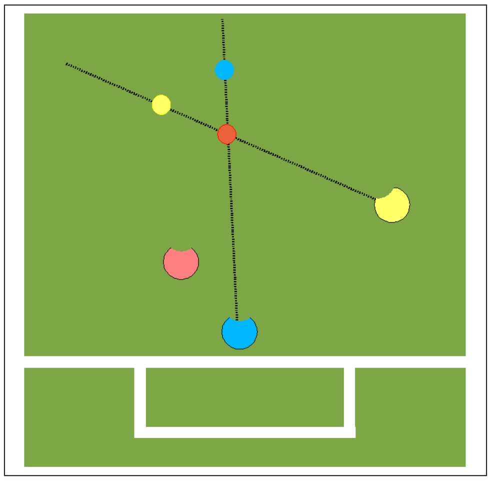


Figure 5.9: The blue robot uses the shared information of the yellow robot to determine a more accurate ball position; the orange ball is the new ball position obtained by triangulation

- Another possibility is to utilise both cameras and try to triangulate the ball position with the different angles at which each camera sees the ball. Although this was implemented, results were very poor since the frontal camera current calibration for the ball angle isn't very accurate, and both cameras position on the robot are very close to each other. This originated the method to return accurate values only when the ball was under 20 cm height and erroneous values if it was above. So until further improvements are made to the vision systems this method is discarded.

Although tests with the first triangulation technique showed promising results, ball triangulation was not utilised in any official game yet. Further tests are required to fully assert the technique reliability and discover any drawbacks that it might bring. This is further discussed in section 9.3 - Future work.

Chapter 6

Roles

As stated in chapter 4 roles reflect given attitudes like striking or defending, and are responsible for selecting the active behaviour at each time step. The decision of a role is based in the current worldstate information and also in the own role internal state.

The roles are responsible for the high-level intelligence of the player. They are one of the key areas of development in the CAMBADA team strategy.

In this chapter improvements made to the existing roles are detailed as well as new developed roles.

When developing a new *Striker* role, it was found that the *Striker* behaviour could be mapped into a structure similar to a hierarchical finite state machine (HFSM). The subsequent role development followed this HFSM like structure as a template for new or modified roles. In the HFSM based structure that was adopted, there are two different transition types:

- *Normal transitions* behave as you would expect. They occur from one state to another triggered by some condition.
- *Global transitions* are somewhat different. The inherent conditions for each global transition are checked in each cycle. Global transitions have a priority order in which they are verified. Should a global condition be verified, the corresponding transition occurs independently of the state the machine is currently in.

In figure 6.1 an example of this type of state machine is shown. In this example, there are several states, normal transitions and three global transitions. The global transitions are checked in priority from left to right. This layout will be followed in figures illustrating the roles state machines (when used).

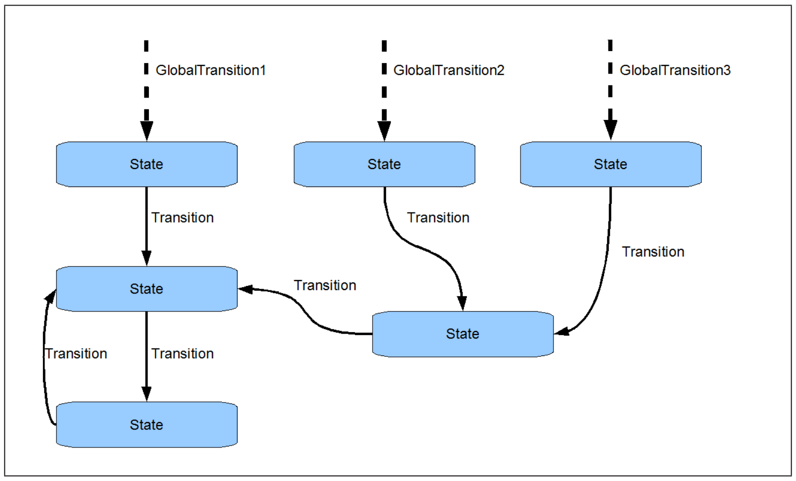


Figure 6.1: State diagram example; global conditions are checked in order

This HFSM structure isn't always present, since there are simpler roles that only execute a few main actions. Although they could be represented by an HSFM with only one or two states, in these cases a more straightforward implementation is used. So in these instances no state diagram is shown.

6.1 Role Striker

The *Striker* role represents the most active and main robot player when the game is in the free play state. It is the robot that is constantly chasing the ball, the only player that is allowed to enter the own or opponent penalty area. If one player has the ball, then that player becomes the *Striker*.

The *Striker* role is intended to:

- Search the ball when not visible;
- Pursue the ball when it isn't in its possession;
- Having the ball, progresse in the field and score;

The *Striker* behaviour is mapped into a state machine that is executed each cycle. This state machine is detailed in figure 6.2. In this section both the transitions and the states will be explained.

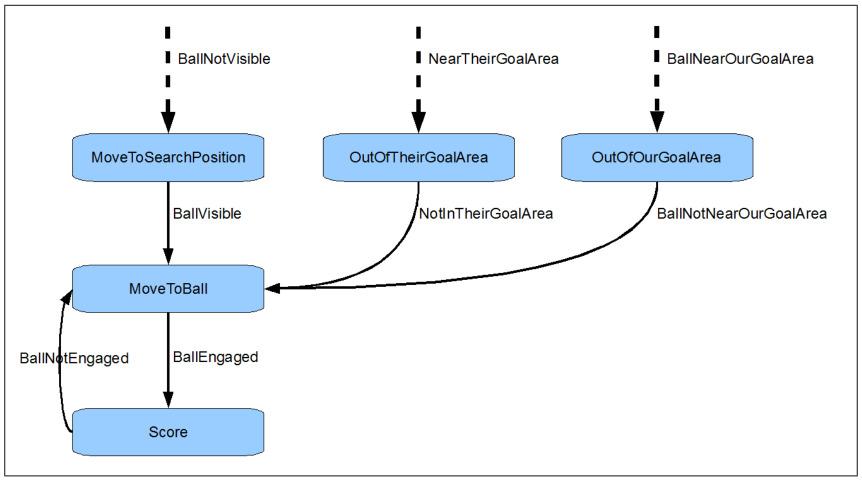


Figure 6.2: *Striker* state machine diagram

6.1.1 MoveToSearchPosition state

This state executes when the ball is not visible to any player in the team. This usually happens when the *Striker* teammates are mainly on their own side of the field, and the ball is far on the opponent team field, which may cause for it not being seen by any of the team players (due to the current ball detection range of 5 meters)[1]. When in this state, the robot will search for the ball. If it is the only field robot running (the goalkeeper doesn't count) it will move to the centre of the field. If there is at least another field robot active, it will progress forward to the opposing team penalty marker position. As the robots have an average of 5.0 meters radius range for spotting the ball, this provides enough field coverage without compromising our defence layout too much.

6.1.2 MoveToBall state

When the ball is visible but not under the *Striker* control, it will move towards the ball and try to gain control of it. A ball position evaluation is executed to verify if the ball is near one of the field outer lines and therefore in danger of going out. This is achieved by executing the *isBallInDangerZone* function [24], which will return the type of danger (if any) or none if the ball is not in such a situation.

If the ball is in danger of going out, instead of moving directly to the ball, the robot moves behind the ball in a way that it will be facing a more advantageous direction when it grabs the ball. This is achieved by calling the worldstate function *getApproachPoint* [24]. The

[1]See section 4.3.2 - Vision

player indicates to which position it should be facing when catching the ball. The function successively returns approach points that will cause the robot to go around the ball and catch it from the outside.

When catching the ball by the opponent goal line, it will try to be facing the field centre. When catching the ball by the sidelines it will try to be facing the opponent goal. This prevents the robot from inadvertently leading the ball out of the field while trying to catch it. It also allows for a faster advance and possible shot as the robot will already be facing the direction in which it is intended to dribble when it catches the ball.

If the ball is not in a danger situation, then a passive interception [2] check is made using the *usePassiveInterception* function, which evaluates if it is useful to use the passive interception to better catch the ball. The evaluation takes into account ball direction, distance, velocity and if it is coming in the current robot direction or not [24]. This is useful because if the ball is indeed moving towards the player, ordering the robot to advance towards the ball will most probably cause it to miss the ball and then have to chase after it, or hitting the ball and not successfully grabbing it, due to the two objects opposing velocities.

If the function check returns true, instead of moving directly to the ball, the *Striker* uses the passive interception behaviour [24], which will cause the robot to move to the ball trajectory closest point, trying to intercept it there, preventing the above mentioned situations in most cases. It also gives the robot a better chance of ending up with the ball. If there is no need to use the passive interception behaviour, then the robot is ordered to move to the ball with the *Move* behaviour.

6.1.3 Score state

In this state, the robot has the ball, so its main objective is to advance in the field and score a goal. In order to achieve this, the *Striker* uses the *worldstate* function *kickable()* that returns the current status of the possible goal opportunity. This function returns five different possible statuses to which the *Striker* reacts accordingly:

- Status: TooFar

  If the distance between the robot and the opponent goal is beyond a certain predetermined threshold, then the status returned is too far, as a shot taken from this distance would pose no real threat to the opponent team. In this case the *Striker* role will execute the dribble behaviour with the direction provided by the utility field[3]. This will

[2]See section 5
[3]See section 5.2

result in the robot coming closer to the opponent goal, with the movement path being based on the predefined information on the utility field while still trying to avoid any obstacles that there might be.

- Status: NotAligned

  This status indicates that the robot isn't aligned with the chosen shooting target. The *Striker* executes the kick behaviour with reduced speed for it to be able to aim swiftly.

- Status: Obstacle

  When this status is returned, it means that there is an obstacle in the first meter ahead of the ball, towards the direction the robot is currently facing. If the robot is at less than 3 meters from the opposing goal, then it executes the kick behaviour, as this can still be a good chance to score. Otherwise it will execute the dribble behaviour to avoid the obstacle and get to a better shooting position with the ball. In this case the dribble is used without the utility field, since the objective is just to avoid the obstacle and try to get closer to the opponent goal.

- Status: DeadAngle

  The robot might be aligned to the target and yet not in a good position to shoot, for example if it is close to one of the opponent corners, the robot could be aimed to the intended target but it wouldn't be in a good position to shoot because the angle opening formed by the two posts would be very small. In order to have a good chance in goal, the robot must have a minimum opening angle between the opponent goal posts. Two symmetrical lines were defined to distinguish which area the robots don't have a high enough opening angle to shoot in goal.

  In order to simplify the problem two lines are used to separate the shooting area from the dead angle area as shown in figure 6.3. This simplifies the math and code involved while providing satisfying results.

- Status: TryToScore

  When this status is returned it means that all the verifications were successful and this is a good opportunity to score a goal, so the *Kick* behaviour is executed, and a shot should follow.

These five different statuses represent the challenges that were found, faced by the *Striker* when advancing with the ball when trying to score. In each status situation different actions are taken to successfully overcome the standing challenges, thus creating a good opportunity to score.

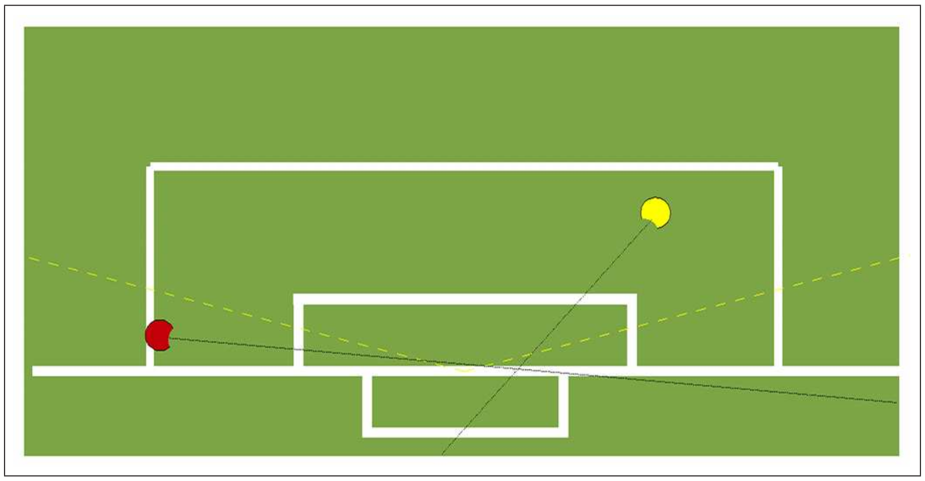


Figure 6.3: The yellow lines delimit the dead angle area; The red robot is in the dead angle area and the yellow robot is not, so the yellow robot can shoot

6.1.4 OutOfTheirGoalArea

In current MSL rules it is illegal for a robot to enter the opponent team goal area. This would result in conceding a free kick to the opposing team. This state exists to detect and prevent such situation.

This state indicates that the robot is near, or inside the opponent team goal area. It usually occurs when the robot is about to enter the goal area. While in this state, there are three possible courses of actions defined:

- If the ball is engaged, it runs the *worldstate shootInGoal()* function, and if it returns true, then the role issues a direct kick command, which will force an immediate kick without any further checks being made, as opposed to using the *Kick* behaviour which would perform its own checks and aiming.

- If the ball is engaged but the *shootInGoal()* check returned negative, the robot will stop as it should be in the borderline of the opponent goal area. It will slowly rotate (roughly 0.6 rad/s) to align itself with the goal centre. The slow rotation is used because a faster rotation would cause the robot to lose control of the ball, as it would roll away. Aligning itself to the goal centre will cause the negative *shootInGoal()* check to become true, resulting in a kick being taken and, usually, in a very good chance to score.

- If the ball is not engaged but it is inside the goal area, then the robot will move to the position in front of the ball, while staying just outside of the goal area, providing an active cover and waiting for an opportunity to regain control of the ball.

6.1.5 OutOfOurGoalArea

In this case the robot will go out of its own penalty area, to avoid causing any illegal defence foul, and to allow the goalkeeper to shoot the ball away.

6.1.6 State machine transitions

The *Striker* state machine global and normal transitions shown in figure 6.2 are explained below.

Normal transitions:

- BallVisible

  When in *SearchBallPosition* state, if the ball becomes visible, either by the *Striker* or other robot (the ball information is shared[4]), this transition is triggered and the active state is changed to *MoveToBall* state.

- BallEngaged

  If the ball becomes engaged for more than 5 consecutive cycles while in the *MoveToBall* state, the state changes to *Score*.

- BallNotEngaged

  If the ball is not engaged for more than 1 consecutive cycle while in the Score state, the state changes to *MoveToBall*.

- NotInTheirGoalArea

  When in the *OutOfTheirGoalArea* state, if the robot comes out of the goal area or if the robot is between the goal and the ball, the state will change to *MoveToBall*.

- BallNotNearOurGoalArea

  When in the *OutOfOurGoalArea* state, if the ball is no longer near our goal area the state transitions to *MoveToBall*.

Global transitions:

- BallNotVisible

  If the ball isn't visible by any of the teammates, this condition is triggered and the current state is changed to *MoveToSearchPosition*.

[4]See section 4.3.1

- isNearTheirGoalArea

  This condition is triggered if the robot is near the opponent goal area. The state is changed to *OutOfTheirGoalArea*.

- isBallNearOurGoalArea

  This happens when the ball is near the own goal area. The active state is changed to *OutOfOurGoalArea*.

These three global conditions are checked in priority order. If one is true the remaining conditions aren't checked.

6.2 Role Midfielder

The CAMBADA team uses a strategic positioning adapted from SBSP and DPRE [26, 27]. When the game is in free play mode, the closest robot to the strategic position number 1, which is 0.5 meters behind the ball, becomes the *Striker*. The remaining field robots will become *Midfielders* and will assume the predefined formation.

The strategic positions are dynamically assigned to each available robot. Determining which free player is the closest to each position. The first positions have higher priority and will be assigned first. Only one position is assigned to each player. The current layout and priorities of the formation used by the team is described in figure 6.4.

The *Midfielder* role is quite simple. It fetches the position number assigned to it by the coach, calculates the corresponding strategic position in the field, and then executes the *MoveToAbs* behaviour to move to that position.

This role was only slightly altered to create one small improvement that results in a more active support of the *Striker* role. If the ball is not between the *Striker* and the opponent goal for more than some predefined time (2 seconds), this usually happens when the *Striker* can't progress with the ball and is engaged with an opponent, then the closest *Midfielder* that is behind the ball also approaches the ball, in order to help the *Striker* acquiring it, or even grab the ball by itself.

This becomes very useful when the *Striker* and an opponent robot are both engaging the ball and none moves forward. In this case, the second robot will help to improve the chances of the team gaining ball possession.

It also provides an advantage when the *Striker* is chasing the ball into its own field which, depending on the ball velocity, can take a reasonable amount of time and will cause the *Striker*

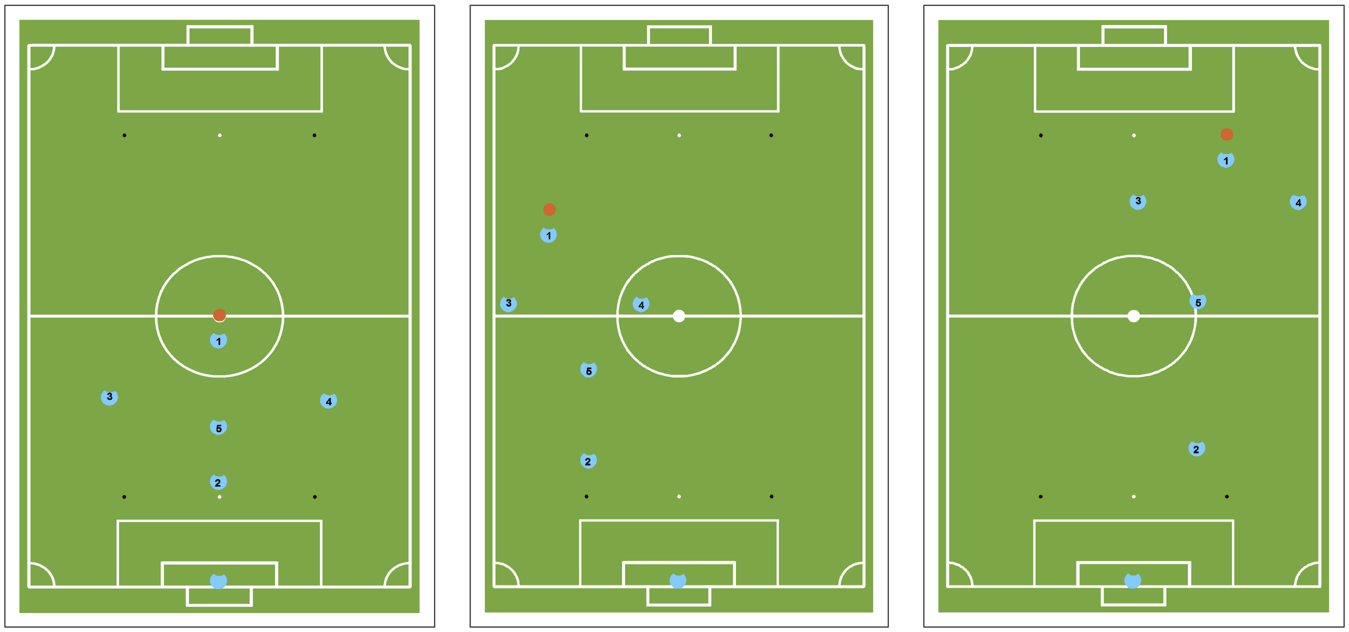


Figure 6.4: *Midfielder* players target positions for some different game situations

to be back in its field when it finally catches the ball. In this case, the active *Midfielder* will catch the ball sooner than what the *Striker* would, and not only will it be further in the field as it will also be better oriented, towards the opponent goal.

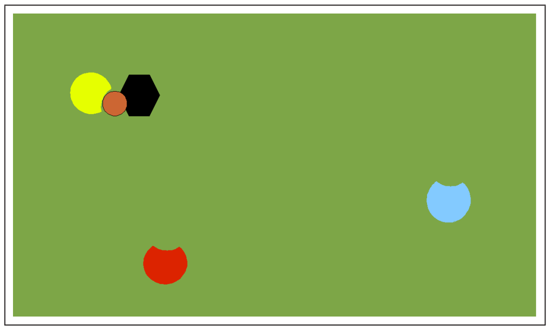


Figure 6.5: Example where a *Midfielder* (in red) would become an active supporter, it is the closest robot to the ball after the *Striker*

6.3 Role supporter

This role was developed to support the *Striker* and to provide a robot that would be in a good position to receive a pass. However, this role is not used in the current team strategy, as the current pass technique only proves useful in a small set of situations, which usually do not occur while the game is in free play.

In section 8.1 a detailed description of the passing technique is given and the reasons that lead to it not being included in the free play game strategy are explained.

To be in a good position to receive a pass, the *Supporter* determines where the *Striker* is, and then positions itself in the opposing side of the field, near the opponent team penalty area. As the *Striker* moves through the field, the *Supporter* actively repositions itself. However, to avoid constant supporter movement when the *Striker* is moving from the left to the right side of the field and vice versa, an hysteresis mechanism was added with a threshold of 2 meters. In this way, only when the *Striker* is beyond 1 meter to the right or left side, will the *Supporter* move to the opposing side.

When the *Striker* is at less than 3 meters from the opponent goal the *Supporter* will fall back and place itself some distance behind the y-coordinate of the *Striker*. This is done because the *Striker* should be about to shoot and moving some meters back causes the *Supporter* to be in a good position to become the new *Striker* in case of a rebound after the *Striker* takes the shot.

6.4 Role Barrier

The *Barrier* role is used to defend set pieces against the CAMBADA team. In these situations all field robots take on the *Barrier* role. Each player determines its own position and moves itself to it.

This defensive positioning is dynamic and calculated based on the ball position. In order for all the robots to come to the same results, all robots must use the same ball position to make the position calculations and assignments. This will guarantee the necessary coherence between players. To use the same ball position, one from the available positions must be chosen. The ball position used is the one given by the robot that is closest to the ball and is able to see it.

The positioning layout and priorities are illustrated in fig 6.6 that shows two possible role *Barrier* layout for an opponent team set piece.

6.4.1 How the positions are determined

After determining the best ball position, one line between it and the centre of the own goal is created. This *barrier line* will be the basis for all the positions in this role.

The first position is the closest to the ball. The robot places itself on the *barrier line*

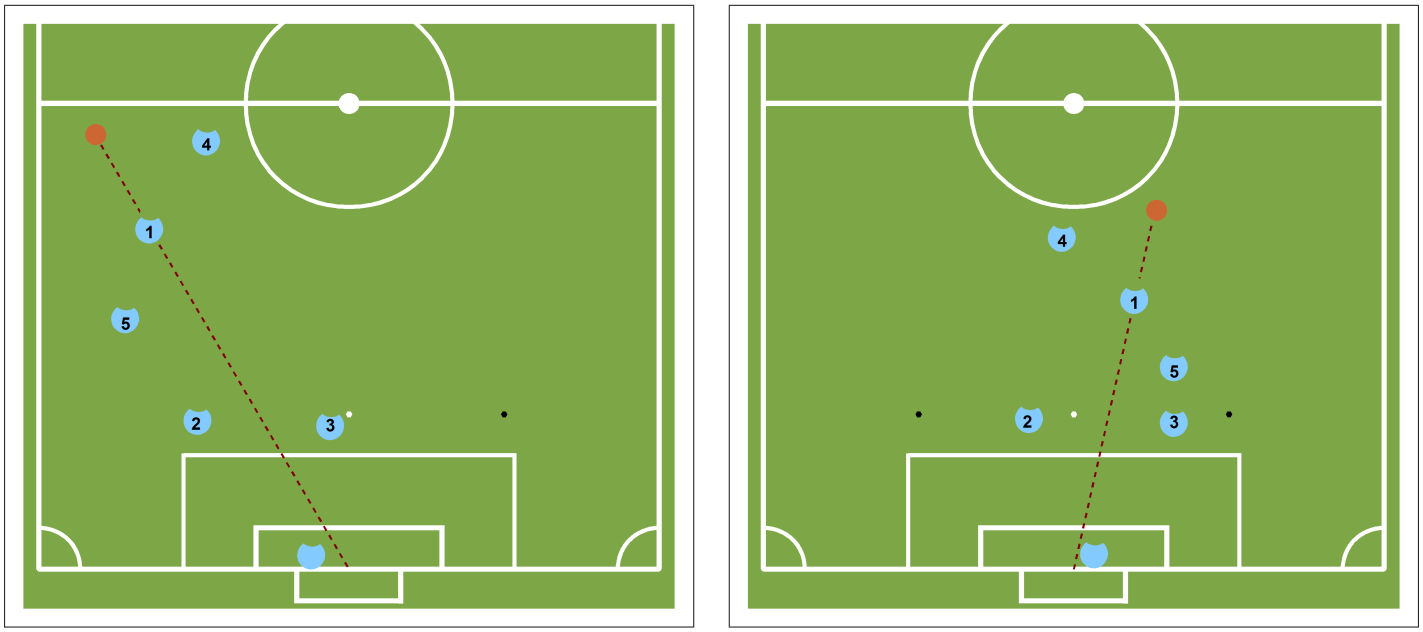


Figure 6.6: Example of role *Barrier* target positions in two different situations; dotted line in red is the barrier line that is the base for the players positioning

2 meters away from the ball or 1 meter in case of being a drop ball foul (these are the minimum allowed distances[5]).

The second and third positions are the defenders that place themselves near their own penalty area. This position is calculated by determining the point on the *barrier line* that is 6 meters away from the ball position and then placing both defenders on each side of that point, with 1.8 meters between them.

The fourth position is the ball watcher. This position is analogous to the first position, with the small difference that it is rotated inwards to the field by 45 degrees. One of the benefits of this position is to prevent faster opponent robots from trying to contour the robot in the first position. An attempt to dribble the ball across to the other side of the field overrunning the first robot is obstructed by this robot.

In MSL games, when scoring a set piece one team can only move after the team scoring the foul touches the ball or after 10 seconds have passed [6]. It is crucial to accurately detect the ball coming into play as soon as it occurs. When it is detected the whole team is notified and the robots will be able to react and move. A quick reaction to the ball coming into play will help in rapidly pressuring the opponent robot holding the ball and prevent it from

[5]See section 2.6 - MSL rules

[6]See section 2.6

shooting.

The main benefit brought by the robot placed in this position is the improvement caused on the team ball movement reaction. This is because this robot is better placed to detect the opposing team robots touching the ball when it comes into play than the robot in the first position. This happens because it is very common for a team taking a set piece to place one robot in front of the ball, this would prevent the rest of team from knowing and promptly reacting to the ball coming into play. By using this position, the teammates are notified sooner, rapidly pressuring the opponent robot holding the ball. This results in increased chances of preventing the opponent team from getting a clear shot on goal.

The fifth position is calculated by placing a robot at 1.5 meters from the first position, rotated 45 degrees to the outside of the field, protecting any eventual attempt from the opponent team on advancing close to the sideline.

After all the positions are calculated, they are tested for erroneous or illegal positioning. Any position that is outside of the field is altered to inside the field and any position inside the penalty area is moved outside. Although it is not illegal to place one robot inside the penalty area, it is illegal to have more than one robot inside it, or one for more than 10 continuous seconds. So it was decided to prevent any *Barrier* position to be inside the penalty area.

6.5 Role Parking

Role *Parking* is used before the game starts, after the set up time, at the end of the first half, and after the game ends. This role orders the robots to “park” themselves on the sideline on a pre chosen quarter of the field, usually where the team is set up. This role takes two main aspects into consideration: how to gather the six robots in positions close to one another, without having them colliding with each other; and also how to make the six of them leave the parked positions without causing a traffic mayhem in the field.

6.5.1 Parking

Each robot can determine its parked position. When the parking signal is sent by the coach the robots will move to a position that is 1 meter inside of the field from the parked position. When they reach that position, they will move to each parked position at a low speed of 0.3 meters per second. This will help prevent possible collisions that would occur from moving all the robots to positions so close to each other.

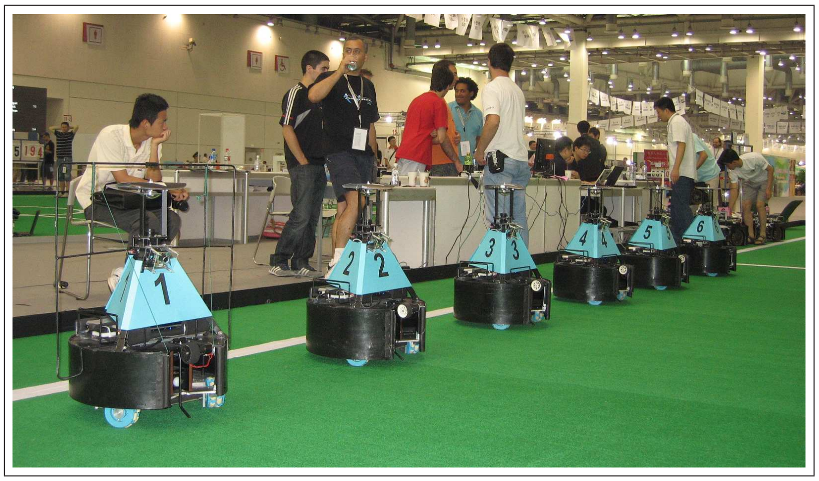


Figure 6.7: Photo of the CAMBADA robots in parking positions

6.5.2 Leaving parked positions

In order to avoid collisions when leaving the parked positions, the robots leave them one at a time, starting with the goal keeper and with a time interval of 1 second between robots. This helps to smooth the team re-entry in the field and provides for a nice coordinated team movement. Also helping the robots leaving the parking position orderly is the method that calculates the parking positions for the goal keeper, which takes into account to where the goalkeeper will have to move next, when leaving the parked position. The method will assign the left most or right most position in accordance to the own goal position, of which the automatic change at half time is accounted for.

6.6 Role Tour

This role was developed for serving as a tool purpose only. It was meant for debug, test,to help tune move behaviours and also to help the vision development tasks by providing a way for the robot to run through a known path continuously.

In this role, a predetermined number of positions are inserted (*waypoints*), which will be followed in succession by the robot, going back to the first after reaching the last one. A position is considered reached after the robot comes within 10 cm of it.

This role was used in RoboCup2008 mandatory challenge with slight modifications. The challenge required one robot to be placed on the field and then, under a timed run, to search and grab the ball, and to try to score with it. The ball used in this challenge was a non colour specific ball, which made the task harder, as the current vision method to detect a non colour specific ball has some limitations, including a smaller range of 2 meters radius. To search for the ball a set of *waypoints* (through all the field) were input and the role ordered the robot to move through them until it spotted the ball. Then a small portion of code extracted from the *Striker* role would take over and would try to grab the ball and score.

The CAMBADA team managed to achieve the 2nd place in this challenge.

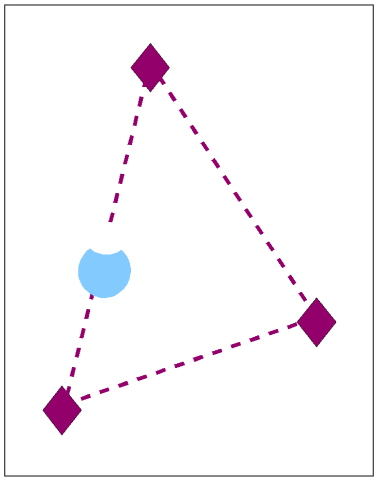


Figure 6.8: Waypoint path example (waypoints in purple); the waypoints will be followed sequentially by the robot

6.7 Role Goalie

The goalkeeper is assigned the fixed *Goalie* role. This role uses the *Goalie* behaviour detailed in the previous chapter, and the *MoveToAbs* behaviour to move the goalkeeper to the goal at the start of the game and also at the half time when the teams exchange sides.

6.7.1 Objectives

This role is responsible for defending the CAMBADA goal. It shares the same objectives any common soccer goalkeeper has, preventing the opponent team from scoring goals.

6.7.2 State machine diagram and conditions

This role follows a simple finite state machine that is illustrated in figure 6.9 and is detailed below:

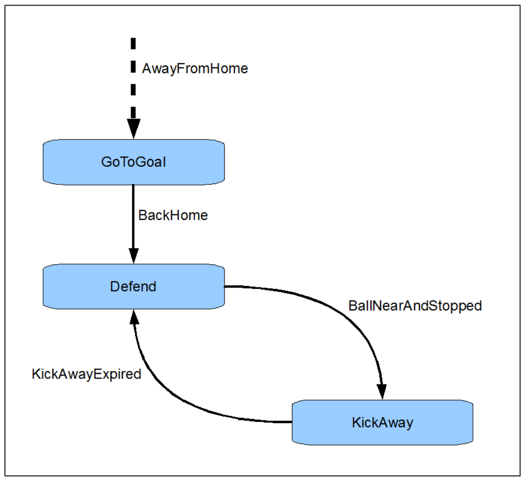


Figure 6.9: *Goalkeeper* state machine diagram

- State Defend

  This is the usual state for the goalkeeper, the only function performed in this state is to execute the *Goalie* behaviour and test for the transition conditions.

- State GoToGoal

  In this state the goalkeeper will execute the *MoveToAbs* behaviour and move to its own goal.

- State KickAway

  This state occurs when the ball is almost stopped (moving at less than 0.2 m/s) in close proximity and in front of the goalkeeper. The goalkeeper will move to the ball and kick it forward with maximum power, clearing the danger. Special attention is given if there is an obstacle in front of the goalkeeper (within 1 meter and at less than 30 degrees from the robot front). In this case the robot moves back to the goal area (while facing the opposite direction and slowly, to maintain ball control) and waits for the obstacle to move away (since given the rules, the robot/obstacle can't maintain its position in the penalty area for more than 10 consecutive seconds).

6.7.3 Transition description

- *AwayFromHome*

  This is a global condition. If the robot is far from what is considered its "home" area, the state will change to *GoToGoal*. The home area of the goalkeeper is approximately the penalty area.

- *BallNearAndStopped*

  When defending, if the ball is near the goal (within a box that spans 2 meters from the goal to the field centre and 1.25 meters to each side) and has been moving slower than 0.2 m/s for more than 1 second. Then the current state will be changed to the *KickAway* state.

- *KickAwayExpired*

  When in the *KickAway* state, if the time in this state is greater than 2 seconds or the ball is not visible anymore, the goalkeeper state will change to *Defend*.

This role development was based on the previous existing *Goalie* role which did not explicitly use a finite state machine but did indeed reflected these three situations. Although not performing the same actions, the general attitude towards each situation was the same.

6.8 Other Roles

There are four more roles: role *Replacer*, role *Toucher*, role *Passer*, role *Receiver*. These roles are made to work in pair, *Replacer* with *Toucher* and Passer with Receiver, due to their coordinated nature they will be explained in the Coordination chapter.

Chapter 7

Coordination

In the RoboCup Middle Size League, autonomous individual robots/agents perform collective actions, to face the existing cooperative challenges in the highly dynamic environment. Coordination and cooperation are vital to achieve success.

To coordinate the several robots actions it becomes very important that all robots share the same playmode obtained by processing the referee orders given through the referee box. The team base station checks the messages received from the referee box, and converts the information received into a state oriented playmode that is broadcasted to robots using the RTDB. This ensures that the delay between the reception of a referee event from the referee box and its awareness by all robots is minimised, enabling a synchronised collective behaviour.

The coordination model of the CAMBADA team is similar to those of other teams. It is based on concepts like *strategic positioning*, *roles* and *formation*. These have been introduced and/or extensively explored in the RoboCupSoccer Simulation League [26, 28].

In this chapter cooperative roles are described. Some cooperative aspects of the implemented strategy are also detailed.

7.1 Making passes

Passing is a coordinated behaviour involving two players, in which one kicks the ball towards the other, so that the other can continue with the ball, possibly in a more advantageous position. Until now, MSL teams have shown limited success in implementing and demonstrating passes. In RoboCup 2004, some teams had already implemented passes, but the functionality was not robust enough to actually be useful in games. In the most recent years some teams occasionally used ingame passes, such as the CoPS team [6] and the Tribots team.

The *Passer* and *Receiver* roles were developed to perform coordinated passes. The *Passer* role uses the *Kick* behaviour, and the *Receiver* role uses the *CatchBall* behaviour to receive the pass. To coordinate between these behaviours a *PassFlag* was created, one for each player. These flags are visible by all the team and can take the following values:

- *None* - The player is not yet ready for any pass coordination
- *Ready* - The *Receiver* is ready to receive the Pass
- *TryingToPass* - The *Passer* is currently trying to pass the ball to the *Receiver*
- *BallPassed* - The *Passer* has kicked the ball towards the *Receiver*

These flags are used in the following manner to cause a coordinated execution of actions by the two players:

RolePasser	RoleReceiver
PassFlag ← TRYING_TO_PASS	
Align to receiver	Align to Passer
	PassFlag ← READY
Kick the ball	
PassFlag ← BALL_PASSED	
Move to next position	Catch ball

Table 7.1: Coordinated actions in a pass

An additional variable was created to indicate to which robot the *Passer* will try to pass the ball. This would be useful in open game play to determine which robot is the *Receiver*. However these roles have not yet been used in open game play, since the current pass technique only allows the ball to successfully be passed at relatively low speeds, approximately 2 m/s. This is the approximate speed at which the robot can progress with the ball, so there is no real gain yet in performing passes in open game play, since making a pass is a risk, and with the opponent fast moving robots, a pass interception would be very likely.

The pass technique developed was demonstrated in Robotica2008 and in RoboCup'2008 MSL Free Technical Challenge. Also a similar mechanism has been used for scoring corner kicks (see below). In the Technical Challenge, two robots alternately took on the roles of *Passer* and *Receiver* while advancing on the field, until one of them was in a position to score a goal. The sequence of actions on both players is as described in Table 7.1 for the pass, with the additional action, performed by the *Passer*, of moving to the next *waypoint* after the pass has been executed. Both players start from their own side of the field and, after each pass, the current passer moves forward in the field to the next *waypoint*, then becoming the receiver of the next pass as detailed in figure 7.1. After receiving the ball at the last *waypoint*, the *Receiver* will execute the *Kick* behaviour and try to score.

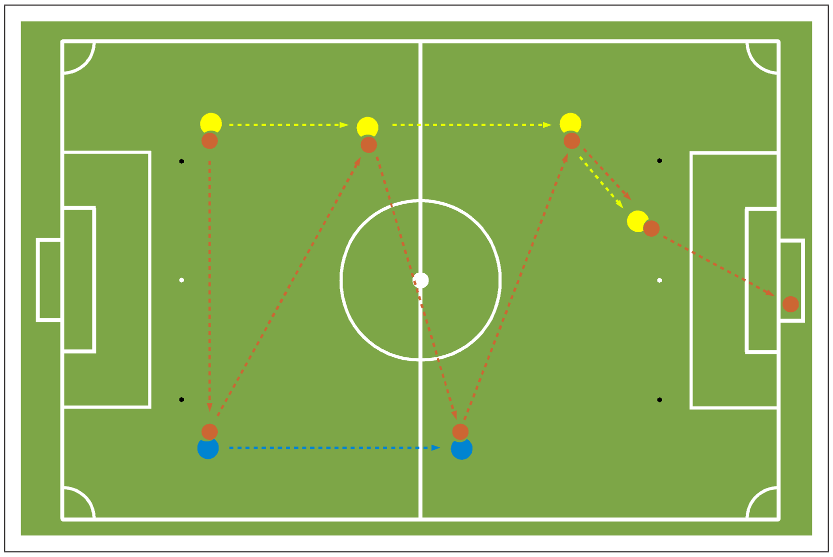


Figure 7.1: Figure detailing the challenge play where two robots exchange the ball to progress in the field and finally score

Small localisation errors and ball trajectory deviations are compensated by the *CatchBall* behaviour (developed in [24]). However, in order for the pass to be executed successfully, coordination between both players is crucial, as it allows the *Passer* to know when the *Receiver* is facing the ball and awaiting the kick and it also allows the *Receiver* to activate the *CatchBall* behaviour after the ball has been kicked, performing then, the necessary backwards movement to absorb the ball impact and successfully catch the ball. It is also necessary for the localisation algorithm to be working properly, as a positional misinformation by one of the robots will cause the *Passer* to kick the ball away from the *Receiver*, and the *Receiver* to expect the ball at the wrong position.

7.2 Set piece strategies

According to the MSL game rules[1], when scoring a set piece, a goal can't be scored directly. Instead, the ball must touch more than one player before entering the opponent

[1]See section 2.6

goal to be validated. To solve this, a technique similar to the one used in real soccer when scoring indirect kicks is employed. This consists in a player touching the ball briefly just for it to come into play, so that the second player can shoot it into goal. This is achieved in the CAMBADA team by the roles *Replacer* and *Toucher*.

This approach using two robots to score the set pieces is similar to those taken by other teams in the MSL, including Tribots, CoPS, Eigen among others.

The *Toucher* is the player that touches the ball into play, and the *Replacer* is the player that grabs and tries to score with it. This is the main concept with which these roles were developed, although in some cases it doesn't exactly reflect what really comes to be.

These roles are used together to perform the set pieces for the CAMBADA team. These are the roles responsible for taking the free kick, throw-in, corner kick, kick-off and goal kick, evaluating and acting accordingly to each of them:

7.2.1 Free kick, throw-in, goal kick

In these situations the robots will align themselves along the line that passes through the ball and the target point to which the *Replacer* will try to shoot (both facing the target point). The *Toucher* will then touch the ball and clear the way so that the *Replacer* can shoot.

Both robots determine the position to which the *Replacer* will try to shoot. This is the same position calculated by the *Kick* behaviour given the current ball position. A line is created based on that target point and the ball position. This is the *shootLine*. The *Replacer* places itself approximately 45 cm behind the ball (this is very close to the ball since the robot has a 25 cm radius and the ball 11 cm radius) facing the target point, and the *Toucher* places itself over the *shootLine*, 45 cm away from the ball (also very close).

After both robots are in place and the referee has given the start command, the game comes into free play mode and the robots will start moving. If it is a free kick or a goal kick, the *Toucher* will move towards the *Replacer*. If it is a throw-in the ball should be placed over the sideline, so instead of being the *Toucher* that pushes the ball, which could cause the ball to go out, the *Replacer* moves towards the *Toucher*.

As soon as the *Replacer* has the ball engaged the *Toucher* (which will know through shared information) will swiftly move away to the side, and after a small period of time (in this case 100 milliseconds) has passed since the ball has been engaged, the *Replacer* will execute the *Kick* behaviour and try to score.

As the alignment has already been made and the robot isn't rotating, this will result in an almost instant shot being taken, therefore not giving the opponents enough time to close in on the *Replacer* and block the shot. This gives the team a very good opportunity to score a goal.

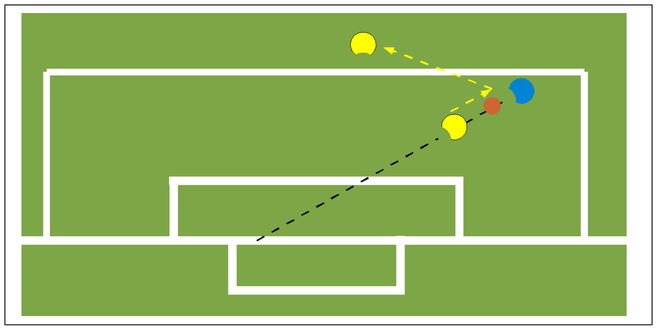


Figure 7.2: The *Toucher* (yellow robot) pushes the ball towards the *Replacer* (blue robot) and then moves away so that the *Replacer* can shoot

This strategy can look inappropriate for the goal kick due to the high distance ($\approx$ 15.3 meters). However due to the effect that the ball bouncing on the field produces visual inaccuracies in most robot vision systems, this causes goalkeepers to be slow to react to these shots or to behave ineffectively resulting in some goals being scored.

Resulting from these far away shots, it is very common for an opponent robot near the opponent penalty area to touch the ball out of the field, or the opponent goalkeeper to defend the ball away, conceding either a throw-in or a corner kick. This is the main benefit of adopting this kicking strategy to the goal kick situation, which tries to create better opportunities to score a goal.

Performing a touch followed by the *Replacer* dribbling the ball to a better shooting position was considered and even implemented. However due to the small velocity at which the CAMBADA robots can move compared to the higher opponent velocities, this technique would usually result in the robot being intercepted before it even left its own side of the field, potentially losing the ball and conceding a good opportunity to the opponent team. Taking all this in consideration the touch and kick strategy was followed also in the goal kick situation as it presented more advantages and less risk over the dribble strategy.

During the Robotica2008 (Aveiro, Portugal) a different approach was also considered for the throw-in situation, in which the *Toucher* would perform a pass to the *Replacer* which would be in the opposite side of the field near the sideline. This strategy proved mildly effective against comparable slow moving teams. However due to the slow speed at which the pass is made, against faster teams (like the ones participating in Robocup 2008 as previously noted) it would be a great risk. A fast moving robot could easily intercept the ball before it reached its intended target receiver. Also considering the high success rate that the indirect kick strategy provided, this pass strategy was substituted by the kick strategy which is currently used in this situation.

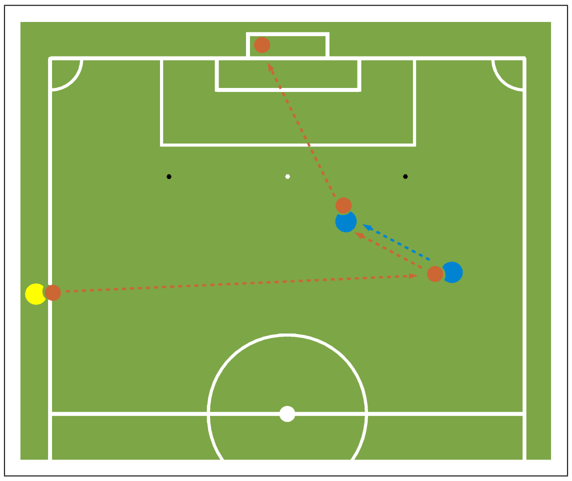


Figure 7.3: The *Toucher* (yellow robot) passes the ball to the *Replacer* (blue robot) that in turn dribbles and shoots to the goal

7.2.2 Kick-off

In this situation a similar approach to the one detailed above is taken. However, since all players must be in their own side of the field at the time of the kick off, the positioning of the robots over the *shootLine* cannot be used. So, both robots place themselves over the midfield line and perform the same strategy as before. In this situation the *Replacer* takes some more time to shoot as it needs to rotate to face the opponent goal, still this results in a reasonably fast movement and often results in a goal being scored.

7.2.3 Corner kick

In the corner kick situation, taking a direct shot is practically impossible since there isn't a good angle to score, therefore new strategies were developed to try and create good opportunities to score. Concerning the corner kick three different strategies were developed:

Touch & Dribble

This strategy consists in a touch being carried out just like in the free kick situation, but instead of the *ShootLine* both robots align themselves to the centre of the field. After the touch is complete, the *Toucher* rapidly moves away and the *Replacer* advances with the ball in the direction of the field centre for a short period of time (≈ 5 s). Upon reaching a favourable position to shoot the *Replacer* executes the *Kick* behaviour which will aim and try to score.

Pass

This strategy consists of making a pass from the *Toucher* to the *Receiver*. To perform this, the *Replacer* moves itself to a position near the midfield and the side line, situated on the same side on which the corner kick is being taken (see figure 7.4). The *Toucher* places itself behind the ball, facing the position in which the *Replacer* is and sets its *PassFlag* to *TryingToPass* (this flag is visible to the other robot).

After the referee gives the start signal, the *Toucher* will grab the ball and execute a pass (low power kick) to the *Replacer* position and will then set the *PassFlag* to *BallPassed*. Upon perceiving the *BallPassed* value in the *Toucher PassFlag*, the *Replacer* will execute the *CatchBall* behaviour which will move slightly backwards to soften the ball contact and will then proceed to catch it. After attaining the ball, the *Replacer* will execute the *Kick* behaviour and try to score.

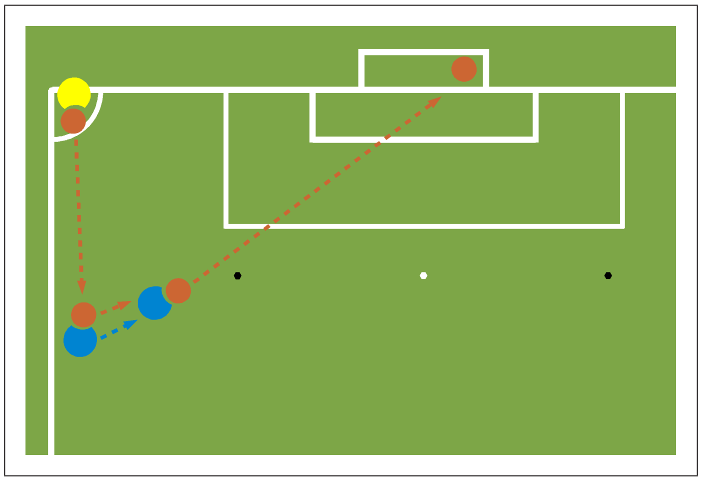


Figure 7.4: The *Toucher* (yellow robot) passes the ball from the corner to the *Replacer* (blue robot) that is in a better position to dribble and shoot to goal

The presence of obstacles between the *Toucher* and the *Replacer* will not currently cause the pass not to be taken. This is intentional. Although usually no team places a robot near the sideline when a corner kick is taken, the situation was considered. The *Toucher* player is near or even over the side line, the *Replacer* is less than one meter away from the sideline. This results in the ball trajectory being very close to the sideline. To intercept this pass a robot must also be placed near the sideline, and due to the current MSL rules at least 2 meters away from the *Toucher*. Since before reaching the *Replacer* the ball still has some speed and also makes some bounces, it would be very difficult for a robot to effectively intercept the pass. Given the ball trajectory, the attempt could easily cause the ball to either fall back to the *Toucher* or bounce on the robot and to outside of the field. Conceding a favourable throw in for the team, which given the location would be a very good opportunity to score a goal.

Cross

This strategy was developed mainly as an alternative to when the pass line would be obstructed in the strategy detailed above. It consists in the *Toucher* making a cross to the opponent Penalty Area, where the *Replacer* would be waiting to try and score.

It was not tested, since the small size of the field currently available to perform development on the CAMBADA project makes this strategy nearly impossible to test. As such, this strategy was also never used.

This approach lost most of its utility due to the decision of not allowing real time decision between which strategy to follow in the corner kicks situation. Both previous strategies will provide a more successful chance of scoring.

The corner kick strategy to be used in each game is chosen prior to the game, by manually setting a parameter in the configuration files loaded to the robots. This method was chosen because it was thought to be more effective to the human part of the team to assess the opponent team and decide which strategy to use in each game, which could also be changed at half time.

7.2.4 Ball search

There is an additional task that is performed by the *Replacer* and the *Toucher*. This happens when the ball position is unknown to all team members, so both the *Replacer* and *Toucher* will search for it. This usually happens when the team players are mainly on their own side of the field, and the ball is far on the opponent side of the field, which may cause for it not being seen by any of the players.

The *Replacer* and the *Toucher* will try to move forward to the opponent field. Each will move near to each side of the opponent penalty area corner, covering great part of the opponent field, and providing a good method to find it.

7.3 Role selection

In the current game strategy there are five roles that are assigned dynamically to the field players. These were detailed in the previous chapter: *Striker, Midfielder, Barrier, Replacer, Toucher.*

At each cycle, each robot will locally execute the role selection algorithm to determine which role is assigned to it. Although it runs locally, it executes based on shared information, in a way that all the robots achieve the same coherent results.

Communication delays are minimal (roughly 100ms) and any possible incoherence is resolved within a few cycles. By analysing the logs from the RoboCup2008 no incoherences in role assignments were found albeit one exception in a game where the team experienced severe localisation and communication problems. In this game there were at instances more than one *Striker* or *Replacer*, but still only for short periods of time.

The role selection method for the field robots can be divided into three modes, according to the current game status: open play, set piece and opponent set piece.

7.3.1 Open play

When in open play there should be one *Striker* and the remaining field robots should be *Midfielders.* The *Striker* will actively try to gain control of the ball while the *Midfielders* will provide passive support accompanying the *Striker* movement according to the predetermined team formation, and occasionally will provide a more active support should it be needed as described in the *Midfielder* functions in the previous chapter. Should the ball move too far away from the *Striker* hopefully a *Midfielder* will be in a good position (nearer to the ball) to become the new *Striker.*

The *Striker* role is assigned to the robot closest to the ball, and the remainder robots take the *Midfielder* role. Should the ball not be visible, the *Midfielders* will layout their formation as if the ball was in the centre of the field and the Striker will go look for it.

7.3.2 Set piece

This happens when one of the following fouls is being scored by the CAMBADA team: kick-off, free kick, goal kick, throw-in or corner kick.

In this case a *Replacer* and *Toucher* are assigned to perform the set piece, while the other field robots are assigned the role of *Midfielder*.

If there is no *Replacer*, the closest to the ball is assigned. Then the same process is used for the next robot closest to the ball which is assigned the role of *Toucher*.

After the role of *Replacer* or *Toucher* is assigned to a robot, this will not change until the set piece is over, no matter which robot is closer to the ball. If because of communication delays or malfunction there happens to be more than one *Replacer* or *Toucher*, the one with the lowest id retains the role while the other gets reassigned.

Should the ball be not visible at the time of the assignment, the closest robots to the *Replacer* and *Toucher* search positions are assigned to each role.

7.3.3 Opponent set piece

This happens when one of the following fouls is being scored by the opposing team: kick-off, free kick, goal kick, throw-in or corner kick, and also occurs in the case of a drop ball situation.

To defend against the opponent set pieces the role *Barrier* is assigned to all field robots.

7.4 Ball triangulation

The current sharing information model utilised by the CAMBADA team is crucial in its current team behaviour, it is of the utmost importance to accurately and efficiently deal with the positioning of the robots in the field, exchange roles, perform the set pieces and even to execute passes.

One additional benefit of the current model is the ability to accurately determine the ball position using more than one robot by performing ball triangulation. This technique takes advantage of the robot localisation ability, the accuracy of each robot in determining the direction of the ball and the sharing of information.

Using two non collinear positioned robots one can easily perform the calculations to accurately pinpoint the ball position and reduce the noise of the ball location when at large distances. This proves very useful to counter the problems of correctly positioning the ball when it is bouncing as described in the previous chapter about the role Goalie, and improves overall accuracy of the ball position.

This technique was described in section 5.4.2 - Bouncing ball, and is further discussed in section 9.3 - Future work.

Chapter 8

Results

The CAMBADA team participated and won the MSL championship in RoboCup'2008 (Suzhou, China, July 2008). Part of the performance evaluation results presented in this section are obtained by analysing log files and videos of games in this championship. In addition, RoboCup'2008 competition results will also be presented.

Due to circumstances, from the end of the RoboCup'2008 Competition to the date in which this work was submitted, the CAMBADA team robots were in China. This prevented the execution of some already planned tests to the performance of some of the work developed during this period. In order to gather results, a software tool was developed to analyse the logs from the RoboCup'2008 games and extract as much relevant evaluation information as possible. The CAMBADA team made it to the final, so it was scheduled to play 13 games. One of them was not played due to absence of the opponent. For two of the games, the log files were lost. The results presented in this chapter are therefore extracted from log files of 10 games.

The logs are created by the coach. Each second it takes a snapshot of relevant information retrieved from each robot. This includes: current role and behaviour, self position, ball position and strategic position. The tool developed took this information into account and extracted some relevant statistics that are discussed in the following sections.

To better understand the game information in the logs, the gameplay should be divided in three modes: free play, set piece for and set piece against CAMBADA. Table 8.1 shows the distribution of time percentage between these modes.

A typical MSL game is roughly half the time in free play mode, in which the teams try to gain control of the ball, progress to the opponent team goal and score. The remaining half of the time is spent in set plays, almost evenly divided between teams.

Game Mode	% of Time
Free play	53.1%
Set piece for	21.5%
Set piece against	25.4%

Table 8.1: Game mode time percentages

This latter percentage of time is considerably higher than what might be expected. This results from the MSL games own dynamics. This is mainly due to the fact of the MSL robots fast moving capabilities (up to 4m/s), which results in the robots being able to cross the field in few seconds. With this move usually ending in a shot being taken, and either scoring or the ball going out. In either case it will result in a set piece, so a high number of set pieces occur per game.

These values are also somewhat inflated due to the time it takes for the robots to position themselves during the set plays, which in some situations can be considerable. Robot substitutions also add to the time inflation. They are made during these periods, in which one team can manually remove one or more of their robots, or insert a robot into play.

From the amount of time spent in set plays for the CAMBADA team it can be perceived why a high importance was attributed to the development of the *Replacer* & *Toucher* combination, which are responsible for scoring the set pieces. A high efficiency rate in the set pieces scoring makes a real difference in the final team performance.

8.1 Roles

In order to give an overview of the CAMBADA players performance in MSL, information regarding role and behaviours distribution during the game was extracted. Table 8.2 shows the average percentage of time any given player spends in each role, with respect to the total time the player is active in each game. Role *Goalie* was not considered because it is fixed to the goalkeeper robot which does not exchange roles, so this table only shows information regarding the remaining five field robots.

It can be observed that players spend a considerable amount of time (45.2 %) in the *Midfielder* role. This is to be expected since when in free play mode there are four players with this role active (the remaining field robot is in *Striker* role), and during the set plays for CAMBADA, the *Midfielder* role is also executed by the non *Replacer* & *Toucher* players.

The *Parking* role amounts for 4.4 % of the time, which occurs just before the game starts,

Role	% of Time
Striker	10.4 ± 5.2
Midfielder	45.2 ± 10.0
Toucher	5.9 ± 4.1
Replacer	5.6 ± 4.6
Barrier	28.4 ± 6.5
Parling	4.4 ± 6.4

Table 8.2: Average times spent by players in different roles (in %) and respective standard deviations

at half time, and after the game ends, albeit slightly higher than expected, it is not unusual for both teams to take some time getting ready. Since this role is not relevant to the team strategy, it will not be considered in statistics from here on.

The remainder of the time is divided between *Barrier*, *Replacer* and *Toucher* roles which are set piece specific, these amount for a combined percentage of 40 % of the time. These values might seem odd, given the results in Table 8.1. However, if we cross reference the roles times with the game modes the information becomes clearer.

The following table shows the role time division percentage across the three game modes:

Roles	Free play	Set piece for	Set piece against
Striker	24.3	0.3	0.4
Midfielder	75.3	51.5	0.3
Replacer	0.0	24.5	0.0
Toucher	0.4	23.7	0.0
Barrier	0.0	0.0	99.3

Table 8.3: Roles Time percentage distribution per Robot in each gameplay mode

From table 8.3 information it can be seen the team reaction to different game situations, as it reflects in the roles attributed to each robot. In free play mode only the *Striker* and the *Midfielder* roles are used. In the set pieces for CAMBADA there are *Replacer*, *Toucher* and *Midfielders*. In the set pieces against CAMBADA all the robots take up the role *Barrier*.

Since there are 5 field robots it would be reasonable to expect a 4 to 1 ratio between *Midfielder* and *Striker* role time percentage in free play mode alone and 2 to 3 ratio between *Replacer*/*Toucher* and *Midfielder* in the set pieces for CAMBADA team. However due to reliability issues, occasional strong collisions and some rare malfunctions there aren't always five running field robots. Table 8.4 shows the percentage of time per number of running field robots.

Nr. of Running Robots	0	1	2	3	4	5
Percentage of Time	0.3	4.5	3.5	16.1	39.3	36.3

Table 8.4: Percentage of time per number of running robots.

With these values in mind it is now easy to understand the lower than expected value for the *Midfielder* role in free play mode. From the log information it was found that the average number of running field robots for the CAMBADA team was 3.98. This reveals the reliability problems that were experienced mostly in the beginning of the competition. These were somewhat sorted out and reliability improved in later games. In the final game the running field robots average was 4.33.

Looking from the team perspective, Table 8.5 shows the roles ratio to each gameplay time.

Roles	Free play	Set piece for	Set piece against
Striker	0.97	0.01	0.02
Midfielder	2.98	2.10	0.01
Replacer	0.0	1.0	0.0
Toucher	0.02	0.97	0.0
Barrier	0.0	0.0	3.97

Table 8.5: Role time per gameplay mode ratio.

This table clearly shows the role assignment priorities in the current CAMBADA team strategy. It can be verified that when in free play mode there is almost always a *Striker*. The small discrepancy of 0.03 is probably due to lag in communications (information sharing) and resulting coordination delays, which would also cause the unexpected 0.02 value in the *Toucher* ratio. This can be understood by the fact that not all robots transition from one game mode to the other simultaneously. This causes small deviations from the expected values. In free play mode there is an average of roughly 3 *Midfielder* supporting the *Striker*. When scoring a set piece for CAMBADA there is always a *Replacer*, accompanied by a *Toucher* as expected. Again the small discrepancy is due to communications delays. As seen before, in set pieces against CAMBADA, all robots are in role *Barrier*.

8.2 Behaviours

Table 8.6 shows the average percentage of time any given player spends running each implemented behaviour. In particular, the second column of the table shows such percentages irrespective of the role taken. The third column shows the percentages of time in each behaviour, considering only the periods in which players are taking the *striker* role.

These values clearly highlight the specific features of the *Striker*: much less time moving to absolute positions, since the *Striker* most of the time ignores its strategic positioning assignment; much more time in moving (to the ball), dribbling and kicking.

Behaviour	% Time (any role)	% Time (Striker)
Move	4.9 ± 3.0	43.7 ± 4.4
MoveToAbs	74.7 ± 12.6	25.3 ± 4.7
Dribble	1.4 ± 1.2	13.4 ± 4.5
Kick	1.8 ± 1.5	14.6 ± 7.7
CatchBall	0.2 ± 0.3	
PassiveInter	0.3 ± 0.2	2.9 ± 1.1
StopRobot	14.7 ± 7.0	

Table 8.6: Average time spent by players running different behaviours (in %) and respective standard deviation

8.3 Coordination

According to the logs, players change roles 2.017 ± 1.022 times per minute. As role assignment is distributed (implicit coordination), it occasionally happens that two players take on the role of *Striker* at the same time. On average, the sum of all inconsistencies in the assignment of the *Striker* role have a combined total duration of 20.9 ± 27.4 seconds per game. The high standard deviation results mainly from one game in which, due to magnetic interference, localisation errors were higher than normal. In that specific game, role inconsistencies occurred 45 times for a combined total of 101 seconds.

Concerning strategic positioning, relevant mainly to *Midfielders*, the average distance of the player to its target position (given by the assigned strategic positioning and the ball position) is 1.381 ± 0.477 m. The strategic positioning assignment for each player is changed on average 7.209 ± 3.335 times per minute.

This somewhat high value and standard deviation might be better understood if the dynamic of a MSL game is taken into account. There are some situations in a game when the ball moves rapidly through the field. This can happen because a fast opponent robot dribbled the ball across the field, or because it shot the ball on goal. During this time, it is very difficult for the CAMBADA robots to quickly compensate the strategic positions variations that accompany the ball fast movement. So the strategic positions assigned to each robot change rapidly with the ball progress, which can explain the higher number of exchanges than what could be expected.

What usually happens in a MSL game is that the CAMBADA robots maintain the same strategic position for some considerable amount of time. When the ball moves quickly through the field, which usually happens for short periods of time, the robots will quickly exchange strategic positions several times to compensate for the ball movement, and then stabilise again. This duality between stable and unstable positioning situations results in the high standard variation observed.

As the CAMBADA players do not track the positions and actions of the opponent players, it is not possible to compute an exact measure of ball possession. However, the game logs enable to compute some related measures, as shown in Table 8.7. The closest player to the ball is at an average distance of 1.2 m from the ball (in a field of 18 ± 12 m). The ball is perceived by at least one player during 91.7% of the time. The ball is engaged in a player grabber device 9.8% of the time.

Average minimum distance to the ball (meters)	1.246 ± 0.325
Time with ball visible (%)	91.7 ± 3.5
Time with ball engaged (%)	9.8 ± 4.7

Table 8.7: Measures related to ball possession.

Figure 8.1 shows the percentage of time the ball was in different regions of the field in the 10 games played by CAMBADA for which we have logs. We see that the ball was in the opponent side for 73 % of time, and that the game was mainly being played in the centre of the field, towards the opponent side.

8.4 Game Detailed Analysis

In the RoboCup'2008 final, CAMBADA faced the Tech United team (European Champions in the German Open 2008). This game was chosen to perform a detailed analysis since it occurred when both teams have had enough time to sort out small bugs or problems, that were experienced in the initial games of RoboCup'2008. It was also chosen due to the high quality

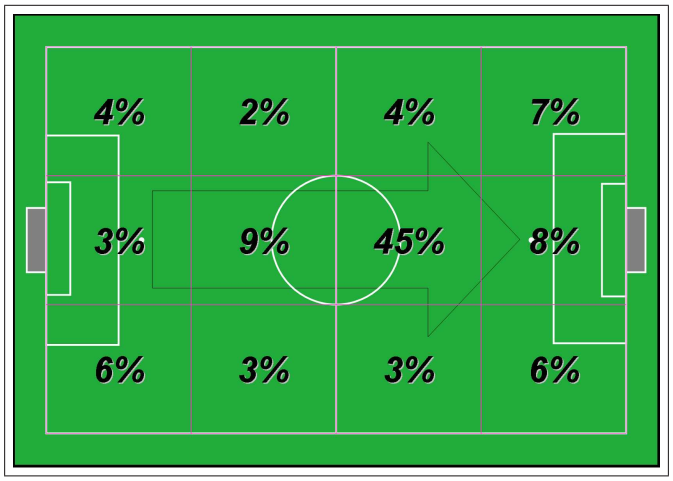


Figure 8.1: Percentage of time the ball was in different locations of the field in 10 games (CAMBADA on the left)

of the opponent team. The Tech United team developed one of the most impressive MSL robots. It has great active ball control with its Innovative Mechatronic Aids. Their robots are very fast, being able to reach 4m/s (the double of what CAMBADA robots currently can). Their goalkeeper has an active frame that can block volley shots. Their robots also have one of the most powerful kicks in MSL (Solenoid Kicker, able to reach 9 m/s).[25]. Considering recent past performance in the German Open in April (1st place), the great robot hardware, and no losses in RoboCup'2008 until the final game, they present a very strong opponent.

Careful analysis of the final game reveals some interesting information regarding the CAMBADA team performance.

Figure 8.2 shows the location in the field from where the ball was shot to goal in the RoboCup'2008 final (CAMBADA-TechUnited).

In the final game, a total number of 21 set pieces for CAMBADA were executed. Table 8.8 indicates the distribution of each type of set piece. The first column indicates the type of set piece, the second indicates the number of occurrences and the third column indicates the number of correctly executed ones.

In 21 set pieces, 18 were correctly executed. The failed throw-in occurred due to magnetic interference in one area of the field which caused the robot to mislocalise itself. The two missed goal kicks occurred because the *Toucher* movement while pushing the ball towards the *Replacer* wasn't accurately aligned and inadvertently pushed the ball into the *Replacer* and out again. This can be due to some small localisation errors experienced near the goal kick marker in the full sized field (versus the training field which has roughly half size). Even

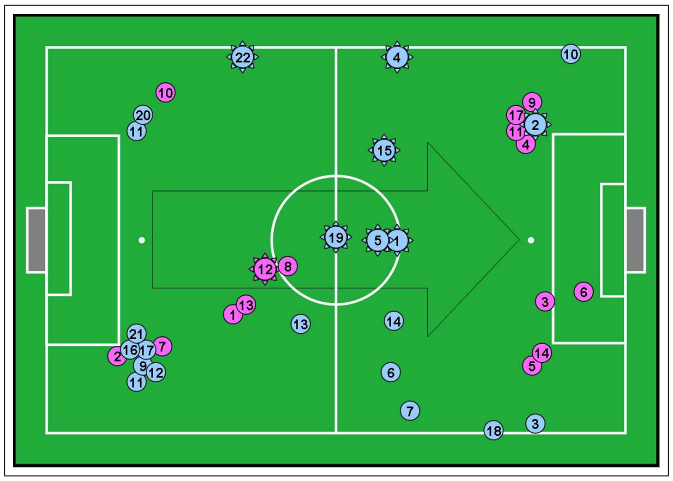


Figure 8.2: Shoot locations in the final CAMBADA (left, blue) - TechUnited (right, magenta) game in RoboCup 2008 (shots are simple circles and goals are sun-like forms)[29]

Set piece	Occurrences	Correct
Kick-off	2	2
Free kick	1	1
Throw-in	6	5
Goal kick	10	8
Corner kick	2	2
Total	21	18 (85.7%)

Table 8.8: Set piece performance

with these three missed set pieces, the successful execution rate was 0.86, which is reasonably high. The three failed set pieces won't be taken into account in the statistics from here on.

In the corner kick situations the chosen strategy was the pass (see Section 7.2.3), and it was successful in the two ocurrences. The pass created two good chances to score, since the receiving robot was in a good position to shoot. Unfortunately the ball velocity in the pass is still quite slow which allowed the opponent fast moving robots to move in and deny a fast shot. The passing technique cannot be improved much further given the current robot kicking ability, which is restricted to volley shots. However based on these evidences, the pass showed promising results. It should be taken into account in future strategies and improved as it can be of great value in future competitions.

The 8 successfully executed goal kicks fulfilled their job reasonably. 4 from the shots missed the target, but the remaining 4 resulted in favourable throw ins being conceded in good positions near the opponent goal. As previously stated in chapter 7, given the current

CAMBADA robot speed, this course of action if preferable to trying to dribble all the way to the other field, and as results indicate, the goal kick situation was transformed in a better chance for goal in half the times. This is a reasonably good result since throw ins are a strong point of the CAMBADA team. Accuracy of these shots should be improved in the future.

The currently considered maximum valid distance from which the CAMBADA robots can accurately shoot is 9 meters (from this distance onwards the shot doesn't have enough power to pose a real threat to the opposing goal). Thus, to analyse the set pieces goal scoring success rate, only set pieces at 9 meters or less from the opponent goal were considered. Corner kicks are not taken into account since a goal can't directly result from its execution. Table 8.9 shows such set pieces and indicates how many resulted in a goal being scored.

Set Piece type	Number	Resulted in Goal
Kick-off	2	2
Free kick	1	0
Throw-in	3	2
Total	6	4

Table 8.9: Set piece efficiency

In the 6 set pieces for CAMBADA that occurred within 9 meters of the opponent goal, 4 resulted in a goal being scored. This is a very good rating of success for the set pieces scored. It can be noted that from the 7 goals scored in this game, 4 resulted from set pieces. This shows the importance of having accurate, reliable and swift set pieces in MSL games. These high values were observed reasonably throughout the whole competition. They were crucial in the team success, proving to be a powerful asset for achieving victories.

The CAMBADA robots also showed great aiming abilities in the competition. Table 8.10 shows the results of all the shots made in the final game within 9 meters of the opponent goal.

Result	Number
Missed	1
Post/Bar	2
Defended	5
Goal	7
Total	15

Table 8.10: Shooting accuracy

A total of 15 shots were made, 1 missed, 1 hit the post and other the bar. The remaining 12 hit the intended target within the goal. This gives an accuracy rating of 80 %. From all the 15 shots made, 7 resulted in a goal being scored. This gives a success at scoring goals (within 9 meters) of 46.7 %. This is an incredibly high value for goal scoring percentage per shot. It is one of the best performing aspects of the CAMBADA team, and one of the reasons the CAMBADA team was the team with more goals scored in the competition.

This high success rate is the result of the accurate ball placing when kicking. CAMBADA scored 7 goals, in 5 of these goals the goalkeeper was actually well positioned and in the path of the ball. However, the accurate calibration and power selection for each kick made the ball reach the opponent goal at an height slightly above 80 cm which effectively caused it to go over the goalkeeper, thus creating a shot that is almost impossible to save (as stated by Tech United themselves [1]).

Table 8.11 presents the competition results of CAMBADA in RoboCup'2008. The team won 11 out of 13 games, scoring a total of 73 goals and suffering only 11 goals.

	♯ games	♯ goal scored	♯ goals suffered	♯ points
Round-robin 1	5	41	2	15
Round-robin 2	4	16	3	9
Round-robin 3	2	5	2	3
Semi-final	1	4	3	3
Final	1	7	1	3
Total	**13**	**73**	**11**	**33**

Table 8.11: Competition results

The CAMBADA team was the best scoring team in RoboCup'2008 with 73 goals scored. This resulted in a 5.6 goal average per game. The next scoring team was the Tribots with 47 goals scored. In the RoboCup'2008 competition CAMBADA achieved the highest number of victories (11, the same as the Tribots team).

The very accurate shots and the good volley shot control, together with fast and successful set piece execution were CAMBADA main assets in scoring goals and defeating the opponent teams. These qualities were crucial in achieving victories and ultimately making the team World Champion.

[1] `http://www.techunited.nl/index2.php?p=news&id=14`

Chapter 9

Conclusion

9.1 Overview of contributions

In this work, some of the behaviours of the robots were improved. This included improvements to the *Move* behaviour rotation issues, a different path planning strategy for the *Dribble* behaviour, redoing the *Goalie* behaviour (contemplating different ball situations) and improving the *Kick* behaviour aiming, target selection and power selection.

Some of the existing roles were rebuilt based on previous work. This was done to contemplate more game situations and to improve general performance.

The *Striker* role was rebuilt. The new *Striker* role is based on a state machine. It becomes easier to understand, test and modify. It also takes into account more game situations than before (ball in danger zone, no ball visible, passive interception, player near goal area) and adequate corresponding actions were implemented.

Role *Midfielder* was added a more active component, as it now interacts more directly with the *Striker* by providing active support in grabbing the ball in a small set of situations.

Role *Supporter* was implemented to provide passive support to the *Striker* and also to act as a backup *Striker*.

The role *Barrier* was created to respond to the opponent set pieces, in which the CAMBADA robots will place themselves in a defending position.

Role *Parking* was created to account for the end of each half of the game, where the robots park themselves near the team setup.

Role *Tour* was created for testing and calibrating purposes, where the robot travels through a predefined route.

Role *Goalie* was rebuilt based on the existing role. It was altered to a state machine which originated simpler programming. The actions for each situation and trigger conditions were mostly redone.

The roles *Passer* and *Receiver* were created to explore the possibility of making a pass between two robots. These coordinated roles allow the CAMBADA team to perform passes for the first time. These roles were used in the RoboCup'2008 mandatory challenge. The *Replacer* and *Toucher* roles were redone and improved in their reliability, accuracy and speed. These roles are responsible for the set piece situations for the CAMBADA team. Several approaches were considered and developed for the different game situations.

The role selection methodology was only slightly altered to include the *Barrier* role, and suffered some minor improvements in the selection code.

Ball triangulation was added to help the goalkeeper to accurately determine the ball position and direction. This technique recurred to the information sharing capabilities of the CAMBADA robots.

9.2 Analysis

The changes in behaviours resulted in some performance improvements in the basic sensorimotor skills of the robots.

The changes in the *Move* behaviour allowed for a slightly faster robot movement, helping the team to respond faster to any situation.

The new path planning used for the dribble behaviour added some flexibility to the way the robot progresses through the field with the ball. It also proved useful as the configuration chosen for the RoboCup'2008 competition allowed the robot to successfully avoid most of the opponent robots that usually concentrated in the middle of the field.

The *Goalie* behaviour was an interesting advancement over the previous one. It provided a more structured approach to each of the considered ball situations and presented proper actions for each of them. In the RoboCup'2008 game it performed reasonably well. However, a faster movement reaction is necessary for better results. This is currently limited by the present motor capabilities of the robot. Improvements in the ball triangulation technique should also help the goalkeeper to position itself more accurately, since sometimes the positioning was somewhat inaccurate.

The improvements made to the *Kick* behaviour proved to be of crucial importance as shown in results from the previous chapter. The high accuracy and general swiftness of the behaviour resulted in one of the strong points of the CAMBADA team. The team success in scoring a high number of goals is in great part due to the work done in this behaviour and its fine calibration that brought great results, as shown in the previous chapter.

The *Striker* role is the main active role of the CAMBADA team. The work done in this role proved successful as the *Striker* correctly adapted to each of the game situations that it usually faces. However there is still considerable room for improvement in the individual actions to be taken in each of the detected situations.

The active support ability of the *Midfielder* showed interesting results. Although the situations which triggered this support were very rare, the *Midfielder* indeed provided a helpful contribution to the active *Striker* robot. Still, it was only a small step in creating a more active support to the *Striker* by the remaining robots. This is definitely an area which should be improved in the future.

The *Barrier* role was a successful addition to the CAMBADA set of roles. It performed as expected by swiftly pressuring the opponent robots in their set pieces. This fast response to opponent movement was greatly aided by the presence of the observer robot which notified the team as soon as the ball came into play. This actively prevented the opponent team from creating good scoring opportunities in most of the situations.

Supporter role was never used or really tested, so as far as it goes, it is only an exploration of an idea and a possible small step into a more complex future role to be developed.

The *Passer* and *Receiver* roles were a very interesting addition to the CAMBADA team, albeit the difficulties present (volley shot only capabilities, reduced ball holding area in the robot). Through good cooperative work by the CAMBADA team, it was possible to successfully develop the passing capability for the players. Although current success is somewhat limited, this is an important step for the team and creates the foundation for future coordination ideas and play strategies. With the pass capability, the new teamwork possibilities are much greater than they previously were.

The *Replacer* and *Toucher* duo became one of the pillars of strength of the CAMBADA team. It was through their effectiveness that the team successfully rose to the first place in the world championship. The fine tuning of these roles with the *Kick* behaviour created a very successful combination that proved to be truly effective in RoboCup'2008. The large amount of time spent in perfecting these roles actions paid off quite well. As shown in the previous chapter they alone are responsible for a good share of the goals scored by the CAMBADA team.

Ball triangulation through information sharing was an interesting idea to counter the bouncing ball effects. However some further improvements are needed to account for the age of the information retrieved from the teammates. Currently the results are somewhat limited and are only of relevant interest when working around the bouncing ball effect. Further work in ball trajectory prediction and alike can improve this technique and its robustness, allowing for it to be used constantly and improve the overall team ball detection precision.

9.3 Future work

This section contains ideas and comments regarding future developments for some key areas that should be developed as well as details on some issues that should be addressed.

The current method for choosing the *Striker* player is not ideal. Currently the player closest to the ball becomes the *Striker*. However it would be more efficient to chose the fastest robot to the ball instead. Taking ball speed into account would avoid certain situations in which the robot is chasing the ball into its own field while another teammate could easily deviate from its position and grab the ball. Also taking the robot own speed and acceleration capability would further help into choosing the more appropriate robot. One possible aspect to be considered is the presence of obstacles between the robot and the ball, which in the end could impossibilitate the robot from catching the ball. Taking all these factors into account and properly calculating the fastest robot to reach the ball, thus making it the striker, would cause a good improvement in the team's responsiveness and would be of crucial importance.

Also regarding the *Striker*, the current path planning strategy is quite limited. Developing a different approach that could assess the current game state (obstacles, teammates) and better choose a path that would avoid interception should improve the team's ability to progress in the field with more success.

The situation in which the ball is between the goalkeeper and the goal line, in which the current action is remain stopped, should be reviewed and a new solution that enables the goalkeeper to safely grab the ball should be developed.

Concerning the *Striker* situation in which it is trying to shoot but it detects an obstacle in the space ahead, a more elaborate action that could enable the *Striker* to successfully dodge the obstacle should be considered.

When kicking into goal, the current shooting point choosing technique is quite limited, as it is only based on the robot own position. A more thorough approach that should analyse the goalkeeper position and best angle to shoot at would significantly increase the goal efficiency.

The ball triangulation technique can be enhanced by using extra information from the teammates. The current information extracted from the teammates (for this technique) is the player position and the relative ball direction. If the relative ball direction angular speed is added, accompanied by the information age, more accurate ball position estimates could be made. With this information, the robot could estimate a more up-to-date relative ball direction for each teammate (and possible error) based on speed and information age. This would result in a great improvement in this method accuracy and reliability. Such improvement could possibly enable the method to be used in most gaming situations and not only for the goalkeeper.

Through game observation it was verified that most of the coordination errors that occurred during the games were due to localisation errors. In order to improve overall teamwork and coordinated team behaviour, the current localisation methods should be reviewed and improved where possible.

The new rules for next year RoboCup edition make most of the current *Replacer* and *Toucher* set piece for free kicks obsolete. The new rules state that the 2nd robot must be 2 meters away from the first robot, so the simple touch strategy will no longer be valid. Therefore the pass approach used in the corner situations should be further developed and implemented for the free kick situations and alike.

The current pass technique has strong limitations, mainly due to the current kicking mechanism that prevents the pass from being faster. With the current speed, the pass has a high chance of being intercepted. It is most likely that the CAMBADA robots will receive a hardware upgrade to the kicking device, allowing straight ground shots. If this happens, the pass technique should be further improved to take advantage of this. The pass efficiency might very well be the next great step for MSL teams. Teams that can master the pass technique and develop fast and reliable passing methods will most likely dominate the following editions of RoboCup and competitions alike.

By rebuilding some of the roles it was observed that a state machine architecture with some small modifications is most likely applicable to most (if not all) roles in the CAMBADA team. To ease the development (testing, creating, modifying) of old and new roles alike, a new abstraction from the c++ code could be created. Developing a tool that could abstract from the current c++ code and help the creation of roles by direct manipulation of a finite state machine could be a major improvement in the team developing capabilities.

The sensing-acting delays in the current architecture were roughly measured to be around 250ms. This delay hinders the reaction capabilities of the robots. Creating a world model that could estimate the ball, opponent and teammates positions could greatly help in the individual robot reaction. If the predicted information is used instead of the outdated information, even with the current behaviours and roles, the robots actions should be more effective and accurate.

Currently only one formation is used by the CAMBADA team, either creating a set of different formations, or a different formation methodology that enable the team to adapt to different situations is of key importance. The different formations could be chosen either by the human members of the team prior to the game, by observing the opponent play strategies, or by the coach software itself, based on the score or some measures of the team efficiency.

When defending, the team relies on the *Striker* role to eventually intercept the ball and opponent player. However a more active defender role should be considered. This role could analyse the game and detect if an opponent is successfully progressing with the ball towards the goal. It could then evaluate the opponent trajectory and plot an intercepting/blocking course, in order to stop or hamper the opponent attacker movement and shoot opportunity.

9.4 Afterthoughts

In the view of the author, the CAMBADA team success was mostly due to the strong teamwork and dedication of the people involved.

The CAMBADA project is a multi-disciplinary venture that required the efforts of several individuals with knowledge in various different areas. It is often said that a team is as strong as its weakest link, well, in robotic soccer it is no different. As detailed in chapter 4, each player is composed of several different and complex hardware components (motors and wheels, cameras, batteries, kicking device, grabber, compass, laptop), these in turn are controlled by several software modules, either in the microcontrollers at the base of the robot, or by processes running in the robot computer. All these software modules communicate and interact. These include monitoring software for each hardware component, visual processing, information filtering, communication handling, information integration, real time reasoning with real world analysis, action planning and coordinated behaviour modules.

The success of CAMBADA (becoming world champion in RoboCup'2008) results from the reliability and value of each of these blocks, and their ability to successfully work together, efficiently carrying out their functions. The constant aim for improvement through reliability and efficiency that the team conveys in its philosophy to its goals and members lead to the successful project that is CAMBADA.

Bibliography

[1] MSL Technical Committee 1997-2008. Middle size robot league and regulations for 2008. http://www.robocup.org, 2007.

[2] Thilo Weigel, Jens steffen Gutmann, Markus Dietl, Er Kleiner, and Bernhard Nebel. Cs-freiburg: Coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, 18, 2002.

[3] Enrico Pagello, Antonio D'Angelo, and Emanuele Menegatti. Cooperation issues and distributed sensing for multi-robot systems. In *Proceedings of the IEEE*, volume 94, No. 7, july 2006.

[4] Brian P. Gerkey and Maja J. Matarić. On role allocation in robocup. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 43–53. Springer, 2004.

[5] Peter Stone and Manuela M. Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. In *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 293–308. Springer-Verlag, 1999.

[6] O. Zweigle, T. Rühr, K. Häussermann, R. Lafrenz, F. Schreiber, A. Tamke, H. Rajaie, A. Burla, M. Schanz, and P. Levi. Cops stuttgart team description 2007, 2007.

[7] Claudio Castelpietra, Luca Iocchi, Daniele Nardi, Maurizio Piaggio, Alessandro Scalzo, and Antonio Sgorbissa. Communication and coordination among heterogeneous mid-size players: Art99. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 86–95. Springer-Verlag, 2001.

[8] Thilo Weigel, Er Kleiner, Florian Diesch, Markus Dietl, Bernhard Nebel, Patrick Stiegeler, Boris Szerbakowski, and Optik Elektronik. Cs freiburg 2001. In *In 5th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences), Lecture Notes in Computer Science*, page 26. Springer, 2001.

[9] K. Yokota, K. Ozaki, N. Watanabe, A. Matsumoto, D. Koyama, T. Ishikawa, Kuniaki Kawabata, Hayato Kaetsu, and Hajime Asama. Uttori united: Cooperative team play based on communication. In *RoboCup-98: Robot Soccer World Cup II*, pages 479–484. Springer-Verlag, 1999.

[10] Oliver Zweigle, Reinhard Lafrenz, Thorsten Buchheim, Hamid Rajaie, Frank Schreiber, and Paul Levi. Cooperative agent behavior based on special interaction nets. In Tamio Arai, Rolf Pfeifer, Tucker Balch, and Hiroshi Yokoi, editors, *Intelligent Autonomous Systems 9*, pages 651–659, 2006.

[11] Hikari Fujii Daiki Sakai and Kazuo Yoshida. Cooperative control method using evaluation information on objective achievement. In R. Alami, H. Asama, and R. Chatila, editors, *DARS04*, pages 201–210, june 2004.

[12] Joaquim Ferreira, Paulo Pedreiras, Luís Almeida, and José Alberto Fonseca. The ftt-can protocol for flexibility in safety-critical systems. *IEEE Micro*, 22(4):46–55, 2002.

[13] EtherCAT Technology Group. Ethercat robots win german open. http://ethercat.org/pdf/english/etg_032008.pdf, 2008.

[14] Roland Hafner, Sascha Lange, Martin Lauer, and Martin Riedmiller. Brainstormers tribots team description. In *RoboCup International Symposium 2008*, 2008. CD Proceedings, Suzhou, China.

[15] M. Oubbati, M. Schanz, T. Buchheim, and P. Levi. Velocity control of an omnidirectional robocup player with recurrent neural networks. In *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Computer Science, pages 691–701, 2006.

[16] Sato Y., S. Yamaguchi, Y. Kitazumi, Y. Ogawa, Y. Yonemura, T. Ueoka, Y. Wada, Y. Takemura, A.A.F. Nassiraei, I. Godler, K. Ishii, and H. Miyamoto. Hibikino-musashi team description paper. In *RoboCup International Symposium 2008*, 2008. CD Proceedings, Suzhou, China.

[17] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L. Seabra Lopes. Coordinating distributed autonomous agents with a real-time database: The CAMBADA project. In *Proc. of the ISCIS*. Springer, 2004.

[18] V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, and J. Fonseca. Implementing a distributed sensing and actuation system: The CAMBADA robots case study. In *Proc. of the 10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2005*, volume 2, 2005.

[19] J.L. Azevedo, B. Cunha, and L. Almeida. Hierarchical distributed architectures for autonomous mobile robots: A case study. In *Proc. of the 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2007*, pages 973–980, 2007.

[20] P. Pedreiras and L. Almeida. Task management for soft real-time applications based on general purpose operating systems. In Pedro Lima, editor, *Robotic Soccer*, pages 598–607, Vienna, Austria, 2007. Itech Education and Publishing.

[21] F. Santos, L. Almeida, P. Pedreiras, L. Seabra Lopes, and T. Facchinetti. An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents. In *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS 2004*, 2004.

[22] F. Santos, G. Corrente, L. Almeida, N. Lau, and L. Seabra Lopes. Selfconfiguration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. In *Proc. of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007*, 2007.

[23] Daniel A. Martins. Image processing system for robotic applications. Master's thesis, Universidade de Aveiro, 2008.

[24] João M. Silva. Sensor fusion and behaviours for the cambada robotic soccer team. Master's thesis, Universidade de Aveiro, 2008.

[25] J.J.M. Lunenburg1 and G. v.d. Ven1 (Eds.). Tech united team description, 2008.

[26] Luís Paulo Reis, Nuno Lau, and Eugénio Costa Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, LNAI 2103*, pages 175–197, 2001.

[27] Luís Paulo Reis and Nuno Lau. Fc portugal team description: Robocup 2000 simulation league champion. In Tucker Balch Peter Stone and Gerhard Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV, LNAI 2019*, pages 29–40. Springer-Verlag, 2001.

[28] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110:241–273, 1999.

[29] Nuno Lau, Luís Seabra Lopes, Gustavo Corrente, and Nelson Filipe. Coordinated action in middle-size robotic soccer. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA2009 (Submitted).*