Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2008

Gustavo Abdul da Fonseca Ussemane Pires Corrente

Arquitectura de controlo/coordenação de uma equipa de Futebol Robótico

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2008

Gustavo Abdul da Fonseca Ussemane Pires Corrente

Arquitectura de controlo/coordenação de uma equipa de Futebol Robótico

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 de Professor Doutor Nuno Lau e Professor Doutor Luís Seabra Lopes, professores auxiliares 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 Luís 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

Agradeço à minha família e amigos por toda a ajuda e incentivos.

Agradeço aos meus orientadores Nuno Lau e Luís Seabra Lopes pela orientação dada.

Agradeço a todos Cambadistas e em especial à malta jovem pelas várias noitadas que nos levaram à conquista do caneco.

A todos aqueles que me ajudaram, o meu obrigado...

palavras-chave RoboCup, Robótica, Sistemas Multi-agente, Coordenação, Cooperação

resumo

CAMBADA é a equipa de futebol robótico da Liga de Robôs Médios da Universidade de Aveiro. Esta equipa foi desenvolvida pelo grupo de investigação de Actividade Transversal de Robótica Inteligente (ATRI), pertencente ao Instituto de Electrónica e Investigação da Universidade de Aveiro (IEETA)).

Este trabalho pretende especificar e implementar uma arquitectura de controlo e coordenação para os robôs CAMBADA. Esta arquitectura é baseada em comportamentos sendo estes utilizados nos papéis de guarda-redes, de médio e de atacante. Este papéis permitiram um evolução do desempenho da equipa CAMBADA nas competições nacionais e internacionais. Foi desenvolvido ainda um mecanismo de posicionamento estratégico, baseado no *Situation Based Strategic Positioning* (SBSP) com *Dynamic Positioning and Role Exchange* (DPRE) da equipa FCPortugal, permitindo maximizar a distribuição dos agente pelo campo. Um treinador, foi implementado com o objectivo de definir as várias posições estratégicas dos agentes CAMBADA.

Este trabalho foi implementado e avaliado ao longo de várias competições nacionais (Robótica 2006, Robótica 2007 e Robótica 2008) e internacionais (RoboCup 2006, RoboCup 2007 e RoboCup 2008). Destas participações é de salientar a vitória em dois campeonatos nacionais, o quinto lugar no RoboCup 2007 e a vitória no RoboCup 2008, campeonato do mundo que decorreu em Suzhou, China.

keywords RoboCup, Robotics, Multi-agent Systems, Coordination, Cooperation

abstract

CAMBADA is a Middle Size League robotic soccer Team from University of Aveiro. This team was developed by ATRI research Group from IEETA.

This work pretends to design and implement an architecture of control and coordination for CAMBADA robots. This architecture is based in behaviors, was them used in roles goal-keeper, midfielder and striker. Those roles allowed an evolution in national and international competitions performance. A strategic positioning was developed based on FCPortugal, Situation Based Strategic Positioning (SBSP) with Dynamic Positioning and Role Exchange (DPRE), maximizing the agent distribution in the field. One coach was developed with the objective to define severals strategic positionings of CAMBADA agents.

This work was implemented and evaluated along severals national (Robótica 2006, Robótica 2007 and Robótica 2008) and international (RoboCup 2006, RoboCup 2007 and RoboCup 2008) competitions. In those competitions is to acclaim the two victories in national competition, the 5th place in RoboCup 2005 and the victory in the RoboCup 2008, world championship placed in Suzhou, China.

Conteúdo

1 Introdução 1
1.1 Enquadramento e Motivação 1
1.2 Objectivos 1
1.3 RoboCup 2
  1.3.1 *Middle Size League* (MSL) 2
1.4 Estrutura da dissertação 3

2 Estado da Arte: Agentes e arquitecturas baseadas em comportamentos 5
2.1 Caracterização do ambiente de actuação (estado do mundo) 5
2.2 Estrutura Física 6
  2.2.1 Forma 6
  2.2.2 Sensores 8
  2.2.3 Actuadores 9
2.3 Tipos de agentes 11
  2.3.1 Agente reactivo: simples 11
  2.3.2 Agente reactivo: com estado interno 13
  2.3.3 Agente deliberativo: orientado por objectivos 13
  2.3.4 Agente deliberativo: orientado por função de utilidade 13
2.4 Estudo de arquitecturas de equipas da MSL 14
  2.4.1 Tribots 14
  2.4.2 COPS 15
2.5 Sumário 15

3 O robô CAMBADA 17
3.1 Base 18
  3.1.1 *Hardware* 19
  3.1.2 Sistema de visão 19

3.1.3 Sistema Computacional 20
3.2 Sumário 20

4 Arquitectura de *Software* da equipa CAMBADA **23**
4.1 Arquitectura de *software* 23
4.2 Arquitectura do agente CAMBADA 26
4.2.1 *Integrator* 26
4.2.2 *Strategy* 27
4.2.3 *Decision* 28
4.3 Gestão da informação do agente CAMBADA 29
4.4 Distribuição de código em sistemas distribuídos 30
4.5 Sumário 31

5 Coordenação da equipa CAMBADA **35**
5.1 Versão 2 35
5.2 Versão 3/4 37
5.3 Sumário 41

6 Conclusão **43**
6.1 Discussão de resultados 43
6.1.1 Síntese do trabalho desenvolvido 43
6.1.2 Artigos publicados 43
6.1.3 Análise de resultados 45
6.1.4 Notas Finais 47
6.2 Trabalho futuro 49

Bibliografia **54**

Anexos **54**

A CAMBADA’2006: Team Description Paper **55**

B ROBOCUP MIDDLE SIZE LEAGUE REFEREE BOX **65**

C An Omnidirectional Vision System for Soccer Robots **73**

D Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots **85**

E CAMBADA’2007: Team Description Paper **97**

F CAMBADA: Information Sharing and Team Coordination **107**

G CAMBADA’2008: Team Description Paper **115**

H The Base Station Application of the CAMBADA Robotic Soccer Team **127**

I Coordinated Action in Middle-Size Robotic Soccer **135**

Lista de Figuras

1.1 Jogo de futebol robótico – final do RoboCup 2008 3

2.1 Exemplos de robôs de forma triangular 7
2.2 Exemplos de robôs de forma quadrada 8
2.3 Exemplos de robôs de forma circular 9
2.4 Exemplos de dois sistema de visão 10
2.5 Sensor de Barreira. 10
2.6 Exemplo de um Sistema holonónico com três motores e rodas omni-direccionais . . 10
2.7 Exemplo de dois sistemas activos de retenção de bola 11
2.8 Arquitectura de base de um agente 12
2.9 Agente reactivo: simples 12
2.10 Agente reactivo: com estado 13
2.11 Diagrama de comportamentos equipa Tribots 14
2.12 Dois exemplos de redes XPlM 15

3.1 Evolução dos robôs CAMBADA 17
3.2 Base 18
3.3 Sensor barreira e bússola electrónica 19
3.4 *Hardware - lógica de controlo* 19
3.5 *Sistema de visão* 20
3.6 Sistema computacional 21

4.1 Arquitectura de *software* 24
4.2 Diagrama de actividade do *Monitor* 24
4.3 Arquitectura do componente *Vision* 25
4.4 Sistema de visão 25
4.5 Arquitectura do agente CAMBADA 26
4.6 Detecção de erros de localização com base na bússola electrónica 27

4.7 Diagrama de estados do *GameState* 28
4.8 Diagrama de Papéis 29
4.9 Diagrama de Comportamentos 30
4.10 Zona de activação do dispositivo de retenção de bola 31
4.11 Diagrama de classes do estado do mundo 31
4.12 Caminhos calculados por A* e A* modificado 32
4.13 Sistemas de coordenadas (absoluto e relativo) utilizado pelo agente CAMBADA . . . 32
4.14 Exemplo de aplicação do *sendToCambadas* 33

5.1 Diagrama classes com os papéis usados na versão 2 36
5.2 Diagramas de estados do papel *Goalie* 36
5.3 Diagramas de estados do papel *Striker* 37
5.4 Formação da barreira 38
5.5 Diagrama de estados do *Striker* 38
5.6 Algoritmo de posicionamento da equipa CAMBADA 40
5.7 Quatro exemplos de posicionamento estratégico 41

6.1 Número médio de golos marcados e sofridos por jogo em competições Nacionais . . 45
6.2 Número médio de golos marcados e sofridos por jogo em competições Internacionais 46
6.3 Localização da bola 47
6.4 Remates executados e sofridos pela equipa CAMBADA 48

Lista de Tabelas

2.1 Comparação de características de vários ambientes 6

6.1 Resultados RoboCup2008 46

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

CAMBADA [1] é a equipa de futebol robótico da Universidade de Aveiro na Liga de Robôs Médios do RoboCup[2]. Este projecto começou em 2003, contando com a participação de investigadores do grupo de Actividade Transversal em Robótica Inteligente (ATRI), do Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA). Os contributos deste grupo incidiram nas áreas de mecânica, electrónica, visão por computador e inteligência artificial.

A equipa CAMBADA participou em vários torneios nacionais e internacionais. Desde 2006, a equipa demonstrou um progresso assinalável no seu desempenho o que lhe permitiu alcançar o 1º lugar no Robótica 2007 e 2008, o 5º lugar no RoboCup 2007 e por último, o 1º lugar no RoboCup 2008 em Suzhou, China.

A equipa CAMBADA é composta por seis robôs que têm como principal objectivo jogar futebol. Este objectivo é complexo devido à complexidade das tarefas e do ambiente dinâmico onde este actuam. Naturalmente, surge a necessidade de desenvolver uma arquitectura de coordenação e controlo para estes agentes. Esta arquitectura deverá dotá-los de comportamentos individuais e fornecer-lhes mecanismos de coordenação.

1.2 Objectivos

O objectivo desta dissertação é a especificação e implementação da arquitectura de controlo do agente CAMBADA, o desenvolvimento e teste sobre essa arquitectura de alguns comportamento essenciais, tais como, o *move*, *dribble* e *kick*, e a definição e implementação de modelos de coordenação, em 2 situações, agentes com capacidade de auto-localização e agentes sem capacidade de auto-localização.

Esta arquitectura deverá ser modular e flexível de modo a permitir uma evolução da equipa. Esta evolução permitirá o desenvolvimento de novos comportamentos e papéis.

---

[1] Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture

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

1.3 RoboCup

O RoboCup [1] é uma entidade sem fins lucrativos, que tem como objectivo principal promover o desenvolvimento da inteligência artificial, robótica e áreas relacionadas. O RoboCup é composto por um Presidente, um conjunto de *Trustees*, um comité executivo e por comités técnicos de cada uma das ligas.

O principal objectivo a longo prazo é: *"No ano de 2050, uma equipa de robôs autónomos humanóides, ser capaz de vencer a equipa campeã do mundo de futebol, num encontro disputado de acordo com as regras da FIFA"*.

Para cumprir o objectivo o RoboCup promove competições e conferências. Assim, pode separar-se as competições do RoboCup nas seguintes categorias:

1. *RoboCupSoccer*
    (a) *Simulation Leagues: 2d, 3d, 3d development, mixed reality*
    (b) *Small Size League*
    (c) *Middle Size League*
    (d) *Four-Legged League*
    (e) *Standard Platform League*
    (f) *Humanoid Leagues: kid-size, teen-size*
2. *RoboCupRescue*
    (a) *Rescue Simulation League*
    (b) *Rescue Robot League*
3. *RoboCup@Home*
4. *RoboCupJunior*
    (a) *Soccer Challenge*
    (b) *Rescue Challenge*
    (c) *Dance Challenge*

1.3.1 *Middle Size League* (MSL)

Na MSL cada equipa é composta por um máximo de seis robôs, cujo tamanho não pode exceder 50x50 cm (na base), 80 cm de altura e 40Kg de peso. Estes robôs devem jogar futebol autonomamente, segundo as regras [2] do RoboCup as quais se baseiam nas regras oficiais da FIFA. O ambiente de actuação é composto por um campo verde (18x12 metros), com linhas brancas. Os corpo dos robôs deverá ser preto e a bola laranja. Cada jogo é disputado em duas partes de 15 minutos.

Durante o desenvolvimento do trabalho desta dissertação as regras sofreram duas grandes alterações. A primeira alteração foi o aumento das dimensões do campo de 12x8 metros para 18x12 metros. A outra grande alteração foi a eliminação da côr amarela e azul das balizas. Actualmente, as balizas são exactamente iguais sem nenhuma côr que as permita distinguir uma da outra.

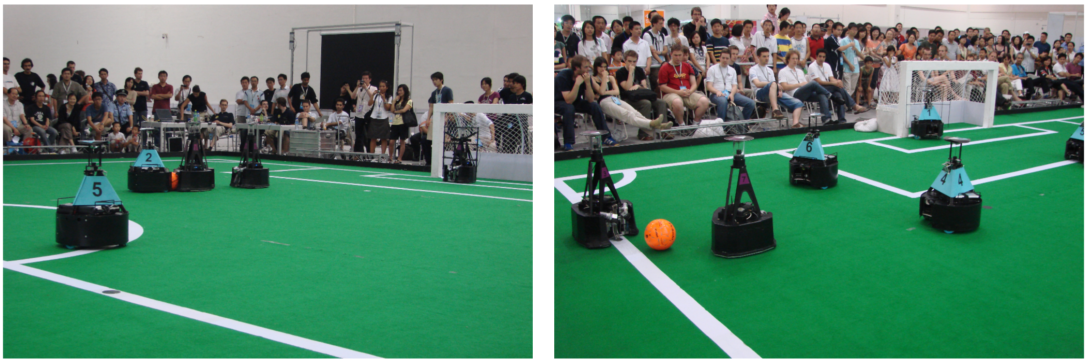


Figura 1.1: Jogo de futebol robótico – final do RoboCup 2008

1.4 Estrutura da dissertação

Esta dissertação está organizada em seis capítulos. O *capítulo 2* descreve alguns conceitos necessários para o enquadramento do trabalho desenvolvido, tais como a definição de agente, do ambiente de actuação, da forma física dos robôs, dos tipos de agentes e da análise de duas equipas chave da MSL. Estes conceitos foram utilizados aquando da especificação e do desenvolvimento da arquitectura e dos mecanismos de coordenação da equipa CAMBADA.

O *capítulo 3* permite visualizar a estrutura física dos robôs CAMBADA. Esta estrutura física é composta por quatro camadas: mecânica, sensorial, de actuação e unidade computacional. Neste capítulo também se apresenta as várias evoluções da estrutura física ao longo do tempo.

No *capítulo 4* descreve-se as especificações da arquitectura de *software* e do agente CAMBADA, também é apresentada a gestão de informação utilizada.

O *capítulo 5* descreve os mecanismos de coordenação utilizados pela equipa CAMBADA. Neste capítulo apresenta-se a evolução dos mecanismos de coordenação ao longo do tempo. Face à evolução do sistema de visão para um visão omni-direccional, foi necessário alterar os mecanismos de coordenação do agente CAMBADA.

O *capítulo 6* apresenta os resultados obtidos no âmbito desta dissertação. Os resultados podem dividir-se em publicações obtidas, resultados de competições e avaliação prática da arquitectura e da coordenação da equipa CAMBADA. Neste capítulo, ainda são apresentadas algumas perspectivas de trabalho futuro, com objectivo de suprimir algumas lacunas da equipa por forma a que esta continue no topo das equipas da MSL.

Capítulo 2

Estado da Arte: Agentes e arquitecturas baseadas em comportamentos

Agente, segundo Stuart Russel and Peter Norvig [3], é uma entidade com capacidade de obter percepções (através de sensores) do ambiente de actuação e de executar acções (através dos actuadores) em função da informação obtida a partir das percepções. Um agente é composto por uma arquitectura que engloba os sensores e actuadores e por um programa. Cabe a este programa processar as percepções recolhidas pelos sensores e gerar comandos que vão ser executados pelos actuadores.

O trabalho da Inteligência Artificial é definir o melhor programa (*software* do agente) possível para executar as melhores acções com a percepção do mundo recolhida, de modo a cumprir os objectivos da melhor forma possível.

2.1 Caracterização do ambiente de actuação (estado do mundo)

O ambiente de actuação pode influenciar a arquitectura do agente visto que cada ambiente pode ter características muito particulares. Uma dessas características é a possibilidade do ambiente ser parcialmente ou completamente observável. Num **ambiente completamente observável** é mais fácil planear as acções a tomar pois tem-se um conhecimento completo a cada nova observação. Pode-se afirmar que se tem uma percepção total do mundo, ou seja, tem-se uma relação de um para um entre a informação recolhida pelos sensores e o nosso estado do mundo. Não é necessário agregar vários tipos informação de modo a gerar outros tipo de informação, só porque não se recebeu nesta observação. Num **ambiente parcialmente observável** não se tem acesso a toda informação do ambiente. Este facto leva à necessidade de manter uma história/memória da informação que vai chegando através dos sensores. Esta história/memória permite manter mais informação a cada instante, pois vai-se agregando toda a informação nova que chega de observação em observação, de modo a melhorar as acções enviadas para os actuadores.

O ambiente pode ser considerado estático ou dinâmico. Quando o agente está a processar a informação observada e ela não se altera, estamos perante um **ambiente estático**. No caso do ambiente se alterar, enquanto o agente está a processar a informação, estamos perante um **ambiente dinâmico**. Alguns autores caracterizam o ambiente também como semi-dinâmico. O ambiente **semi-dinâmico** é um ambiente que não se altera durante o processamento da informação por parte do agente, mas esse tempo

Ambiente	Observação	Estático	Discreto	Sequencial	agente
Xadrez	Completa	Estático	Discreto	Sequencial	multi
Xadrez com relógio	Completa	Semi-estático	Discreto	Sequencial	multi
Resolver um puzzle	Completa	Estático	Discreto	Sequencial	mono
Controlador	Completa	Dinâmico	Contínuo	Sequencial	mono
RoboCup	Parcial	Dinâmico	Contínuo	Sequencial	multi

Tabela 2.1: Comparação de características de vários ambientes

de processamento influencia directamente o desempenho das acções tomadas que serão enviadas para os actuadores.

O número de estados que o ambiente pode ter, é outra característica importante. Quando o ambiente tem um número finito de estados possíveis, diz-se que é um **ambiente discreto**. No caso do ambiente tem um número infinito de estados, considera-se que é um **ambiente contínuo**.

Uma característica que importa referir é o facto do ambiente poder ser sequencial ou por acontecimentos. Num **ambiente sequencial**, cada ciclo de percepção/processamento/acção influencia o próximo ciclo. Já num **ambiente baseado em acontecimentos/episódios** qualquer que seja a acção tomada, não influencia o próximo ciclo.

O ambiente de actuação pode ser caracterizado pelo número de agentes. Assim sendo, pode-se dizer que o ambiente é mono agente (*single agent*) ou multi agente. Num **ambiente mono agente** existe apenas um agente a actuar sobre o mundo. Já num ambiente **multi agente** temos mais de dois agentes a actuar sobre o mundo. Num ambiente multi agente, os vários agentes podem competir entre si ou cooperar entre si.

O ambiente de actuação dos robôs futebolistas da MSL é um ambiente muito complexo, como se pode visualizar na tabela 2.1. Esta complexidade deve-se ao facto do ambiente ser parcialmente observável, com multi agentes que cooperam entre si (os robôs da mesma equipa) e competem entre si (os robôs adversários). Como o ambiente é o mundo real, é um ambiente contínuo e dinâmico.

2.2 Estrutura Física

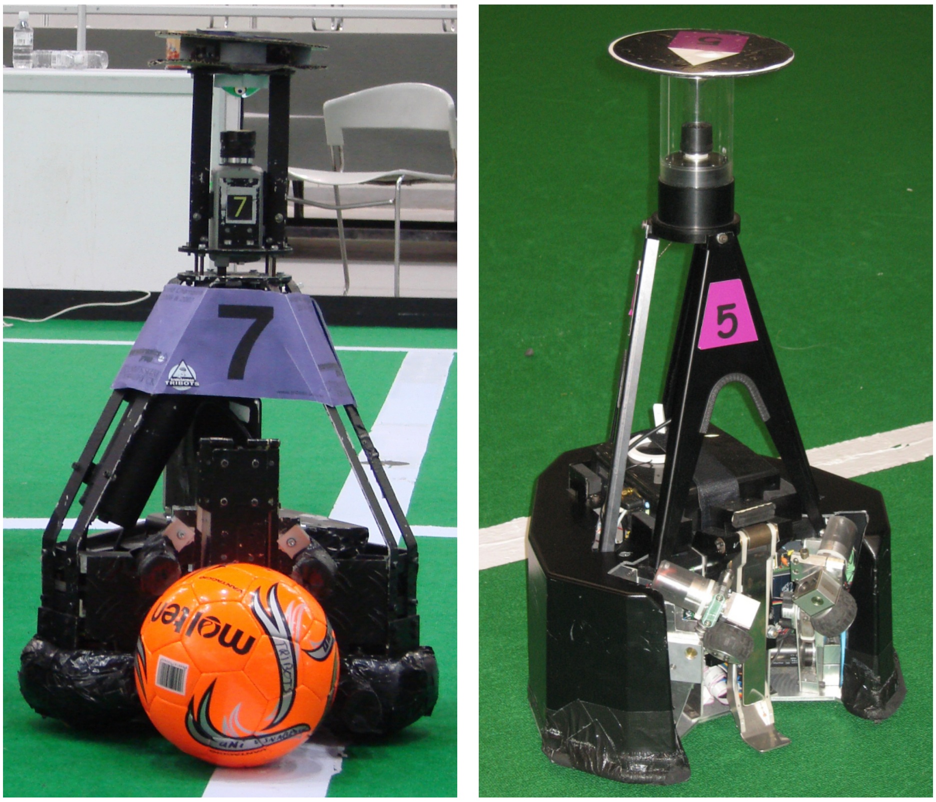
Um agente não tem de ter necessariamente uma estrutura física/mecânica. Contudo, como este estudo se enquadra no âmbito do RoboCup, mais em concreto na MSL, na secção 2.2.1 pretende-se caracterizar a parte física dos robôs usados nesta liga.

2.2.1 Forma

As equipas do futebol robótico na MSL apresentam as mais diversas formas nos seus robôs, podendo os mesmo ter a forma triangular, quadrada, circular ou mesmo hexagonal.

Triangular / Quadrada

Na figura 2.1 visualiza-se dois exemplo de robôs com forma triangular [4–6], já na figura 2.2 pode visualizar-se dois robôs com forma quadrada [7]. Estas duas formas de robôs têm como principal



(a) Robô da Equipa Tribots (b) Robô da equipa TechUnited

Figura 2.1: Exemplos de robôs de forma triangular

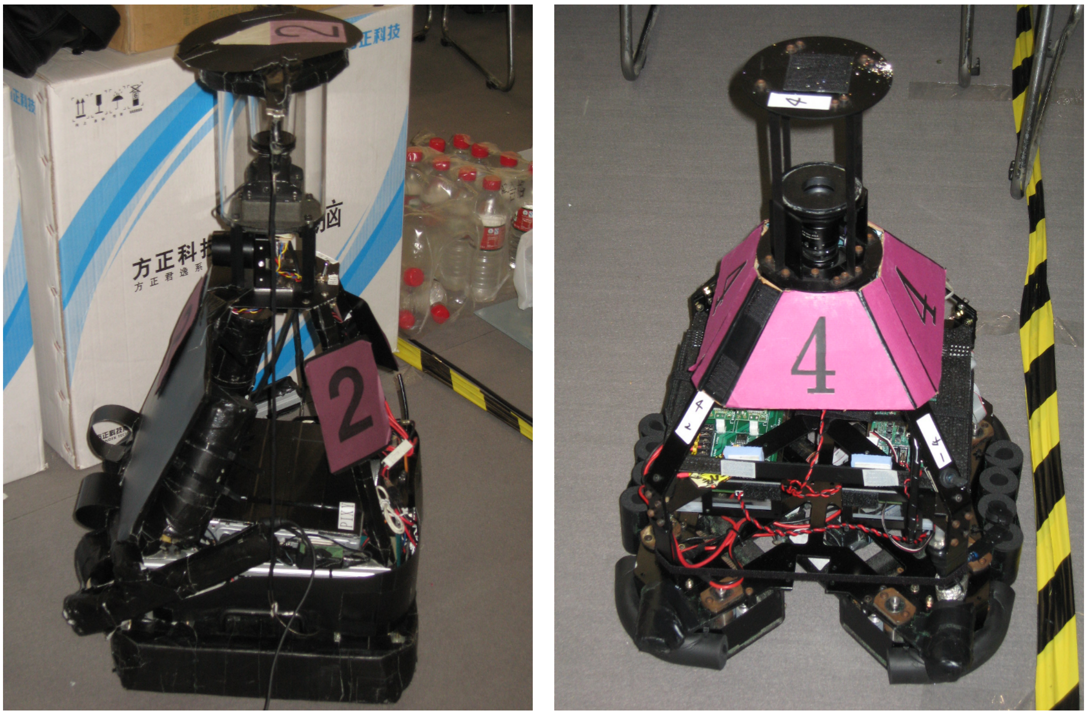
vantagem, permitir tocar/controlar a bola com qualquer uma das faces. A principal desvantagem que esta forma apresenta é a sua orientação poder ser influenciada caso um robô adversário lhe toque num dos vértices. Esta situação pode ocorrer quando o robô conduz a bola ou no momento de remate. Outra desvantagem, é quando numa situação de disputa de bola, o robô com estas duas formas não conseguem rodar, o que os impede de a disputar nas melhores condições.

Circular

A forma circular foi a forma adoptada pelas equipas portuguesas, CAMBADA e MINHO[1]. A principal vantagem deste tipo de solução é o robô poder rodar sobre si próprio, mesmo que esteja rodeado por outros robôs. A principal desvantagem é não poder tocar/dominar a bola com qualquer ponto do ser corpo, de uma forma precisa e exacta.

---

[1] `http://www.robotica.dei.uminho.pt/robocup`



(a) Robô da equipa Eigen (b) Robô da equipa TKU

Figura 2.2: Exemplos de robôs de forma quadrada

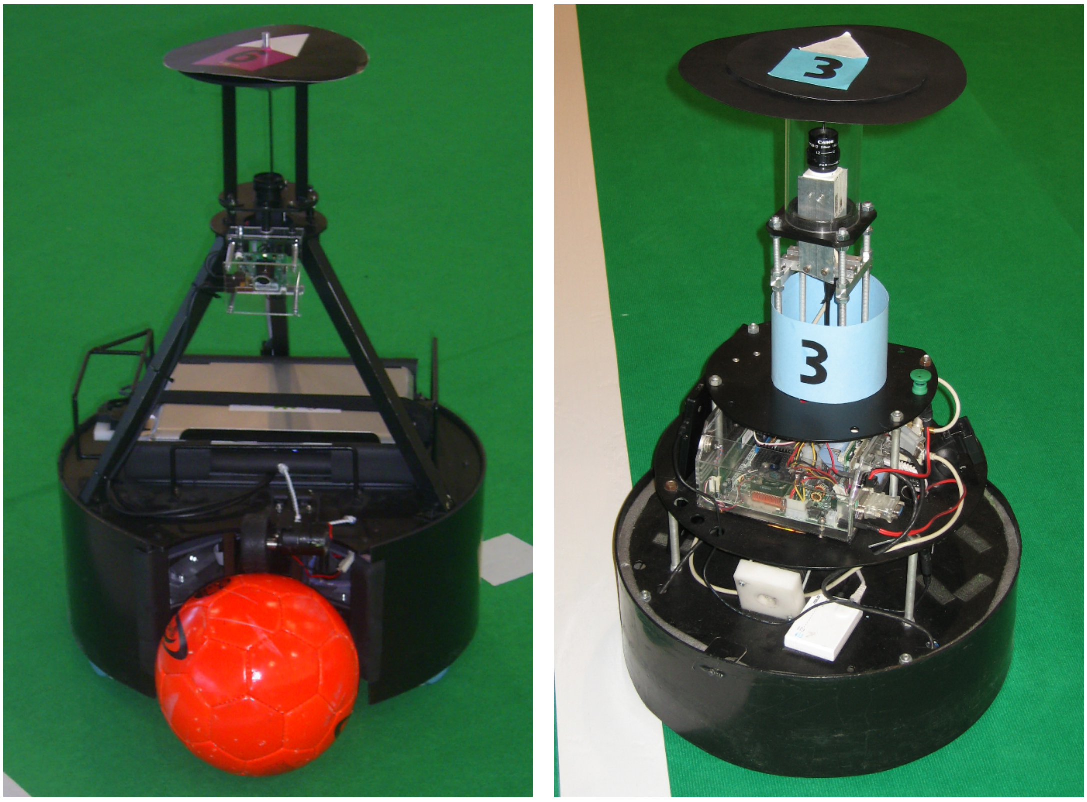
2.2.2 Sensores

Visão

As câmaras são utilizadas por todas as equipas como sensores principais. Uma visão omni-direccional é conseguida através de um sistema catadióptrico [8]. Este tipo de sistema é composto por um espelho cónico (hiperbólico ou parabólico) e por uma câmara, como se pode ver na figura 2.4(a). Existem equipas que preferem não usar um sistema catadióptrico. Estas usam apenas câmaras [9, 10] que olham directamente para o ambiente. Com o objectivo de recolher o máximo de informação possível, neste tipo de solução é normal utilizar-se mais do que uma câmara, sem estar fixa, permitindo olhar para vários locais (ver figura 2.4(b)).

Orientação

A bússola electrónica é um sensor de orientação que é utilizado por algumas equipas como, por exemplo, as equipas CAMBADA e TechUnited [5, 11]. A necessidade do uso deste sensor surgiu devido ao desaparecimento das cores das balizas. O uso deste sensor é útil para desambiguar aquando da auto-localização no campo de jogo.



(a) Robô da equipa CAMBADA (b) Robô da equipa MINHO

Figura 2.3: Exemplos de robôs de forma circular

Sensor de barreira

Sensor de barreira é um sensor (ver figura 2.5) utilizado pelas equipas CAMBADA [11] e MINHO. Este sensor permite detectar quando é que a bola está encaixada na frente do robô.

2.2.3 Actuadores

Os robôs da MSL apresentam vários tipos actuadores: sistema de tracção/motores, sistema defesa do guarda-redes e os dispositivos para reter e chutar a bola.

Sistema de tracção/motores

Os motores são os principais actuadores, pois permitem o deslocamento dos robôs. Os sistemas de tracção podem dividir-se em dois grupos: os holonómicos e os diferenciais.

O sistema de tracção holonómico permite que o robô se movimente para qualquer direcção com qualquer orientação. Para implementar este tipo de sistema são utilizados vários tipos de configurações. A primeira configuração, e a mais usada, é de três motores a 120^o [4, 11, 12], usando rodas omnidireccionais como se pode ver na figura 2.6. A segunda configuração é a utilização de quatro motores/rodas dispostas a 90^o.

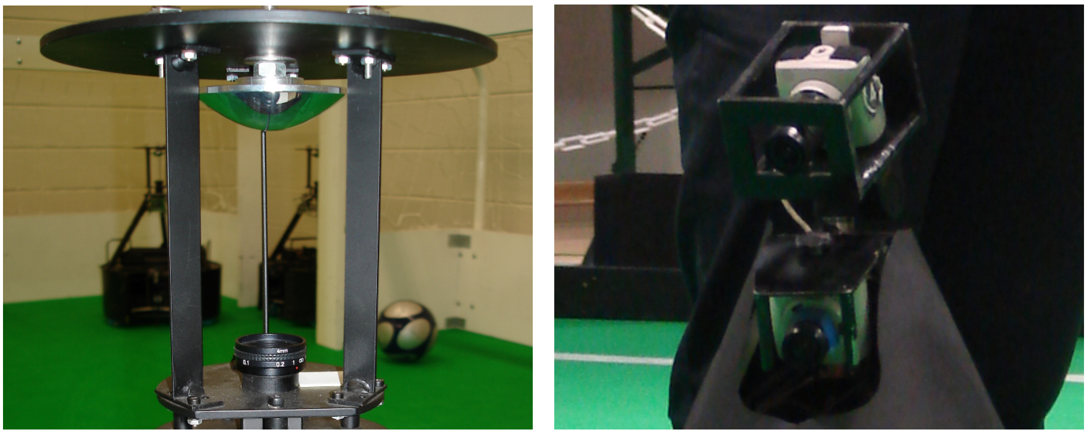

(a) Sistema de visão catadióptrico (b) Sistema de Visão não catadioptrico

Figura 2.4: Exemplos de dois sistema de visão

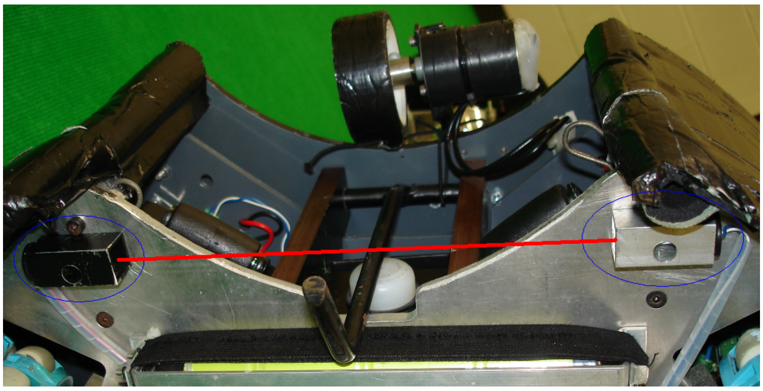


Figura 2.5: Sensor de Barreira.

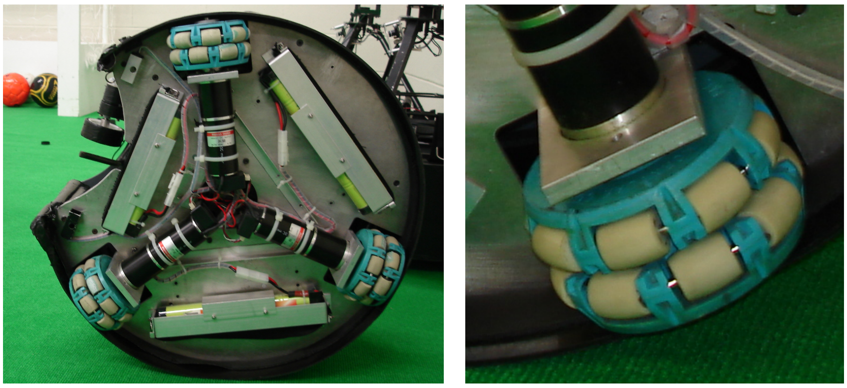


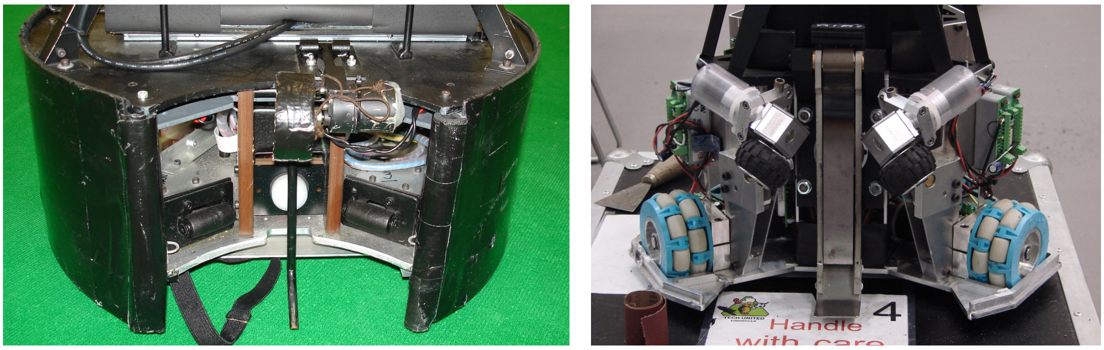
Figura 2.6: Exemplo de um Sistema holonónico com três motores e rodas omni-direccionais

Dispositivo de retenção de bola

Os dispositivos para retenção de bola dividem-se em 2 grupos, os dispositivos passivos e os dispositivos activos.

Nos dispositivos passivos a implementação mais comum é o uso de uns ”dedos” que podem ser de esponja to de borracha. Estes ”dedos” têm como objectivo amortecer a bola aquando a sua recepção e permitir também guiar a bola durante a sua condução.

Os dispositivos activos de retenção de bola [5], como o próprio nome indica, têm um mecanismo activo que permite controlar a bola. Este mecanismo normalmente é implementado usando um motor e um roda. Na figura 2.7 pode-se visualizar dois exemplos de sistemas activos de retenção de bola.



(a) Equipa CAMBADA (b) Equipa TechUnited

Figura 2.7: Exemplo de dois sistemas activos de retenção de bola

Dispositivo de chuto

Este dispositivo é um dos mais importantes, pois vai permitir chutar a bola e consequentemente marcar golos. Existem três tipos de dispositivos de chuto: os electromagnéticos [5], os hidráulicos e os baseados em mola [13].

Os dispositivos de chuto electromagnéticos e hidráulicos têm como principal diferença em relação aos dispositivos baseados em mola, maior facilidade de dosear a força de actuação, ou seja, escolher a força de chuto.

2.3 Tipos de agentes

A missão da Inteligência Artificial, como o próprio nome indica é dotar os agentes de inteligência para cumprirem as suas tarefas de uma forma eficiente e autónoma.

A figura 2.8 permite contextualizar o papel da Inteligência Artificial na arquitectura de um agente. Assim sendo, a sua missão é definir o *centro de decisão*. Este centro de decisão pode ser implementado segundo vários modelos, designadamente: agente reactivo simples, agente reactivo com estado interno, agente deliberativo orientado por objectivos e agente deliberativo orientado por função de utilidade.

2.3.1 Agente reactivo: simples

Os agentes reactivos simples, como o próprio nome indica, reagem às suas percepções do ambiente. As acções escolhidas são sempre com base na sua percepção actual sem utilizar informação de uma

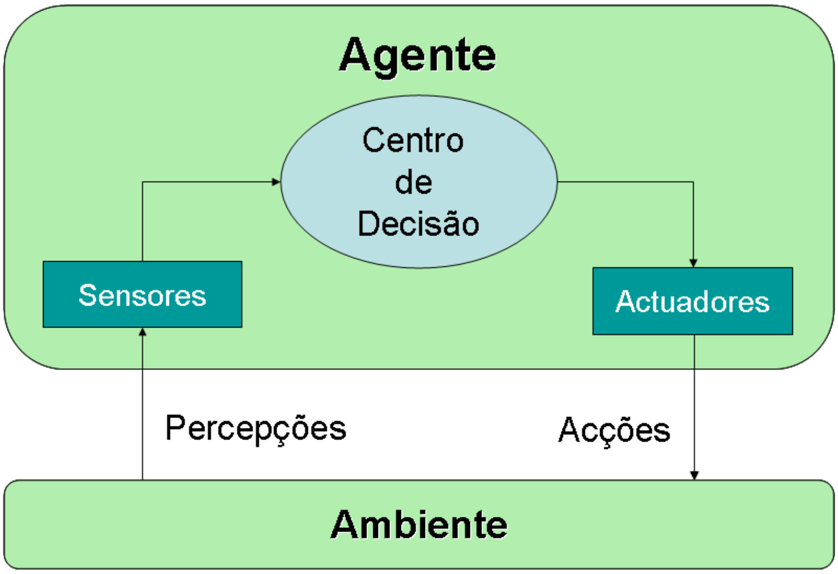

Agente
Centro de Decisão
Sensores
Actuadores
Percepções
Acções
Ambiente

Figura 2.8: Arquitectura de base de um agente

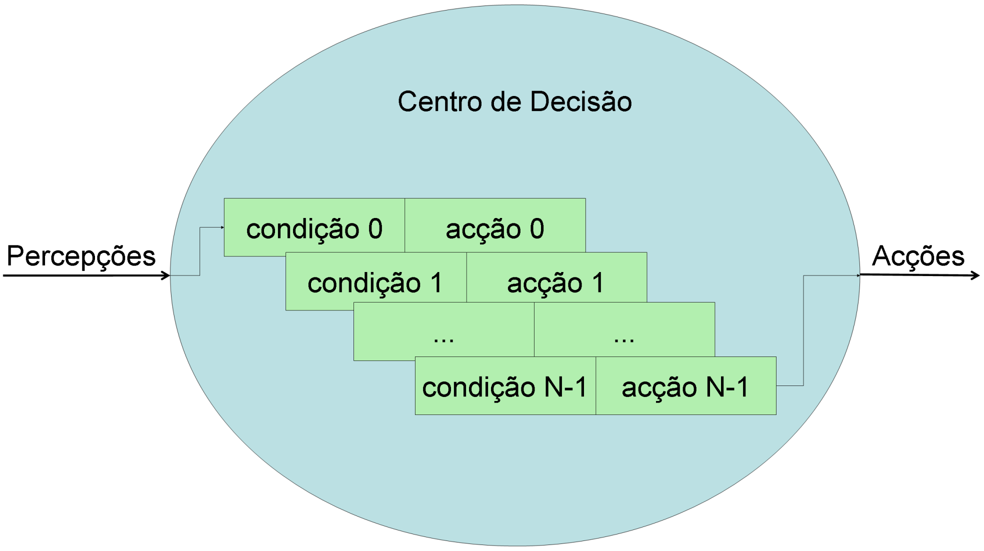

Centro de Decisão
Percepções
condição 0
acção 0
condição 1
acção 1
...
...
condição N-1
acção N-1
Acções

Figura 2.9: Agente reactivo: simples

percepção passada, ou seja, não há história de percepções.

A implementação deste tipo de agente tem como base um conjunto de condições (*ver figura 2.9*) que perante um certo valor de uma percepção resulta na escolha de uma determinada acção. Esta escolha é feita de forma sequencial, ou seja, vai-se percorrendo as condições e quando se encontrar uma condição verdadeira, escolhe-se a acção associada a essa condição.

Este tipo de agente, como não mantém um estado interno, é susceptível de ficar a oscilar entre duas ou mais condições. Uma solução para resolver este problema é introduzir um factor aleatório no momento de decidir a acção a executar.

2.3.2 Agente reactivo: com estado interno

Os agentes reactivos com estado interno têm um funcionamento muito semelhante aos agentes reactivos simples apresentados na secção 2.3.1, mas têm uma diferença, mantém um estado interno. Este estado interno vai permitir acumular o conhecimento do ambiente recolhido em percepções anteriores.

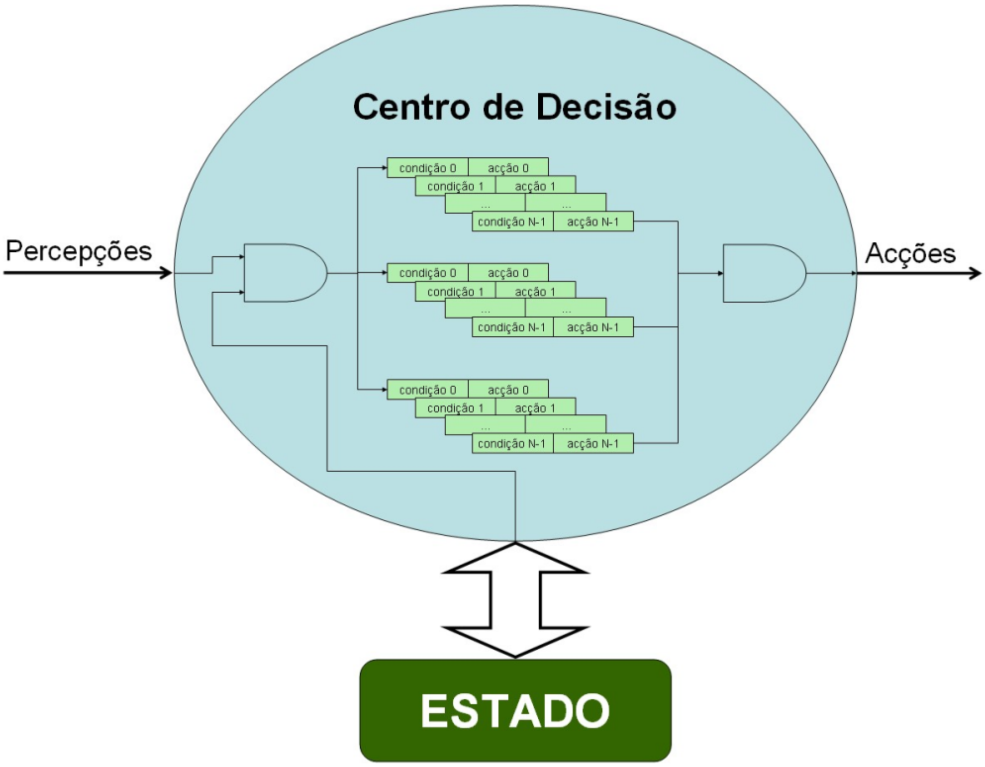

Centro de Decisão
Percepções
condição 0
acção 0
condição 1
acção 1
...
...
condição N-1
acção N-1
condição 0
acção 0
condição 1
acção 1
...
...
condição N-1
acção N-1
condição 0
acção 0
condição 1
acção 1
...
...
condição N-1
acção N-1
Acções
ESTADO

Figura 2.10: Agente reactivo: com estado

Na figura 2.10 pode ver-se que é possível ter um conjunto de condição-acção por estado. Neste tipo de agentes cada condição pode mudar o estado interno do agente, permitindo mudar de estado e consequentemente passar para outro conjunto de condição-acção.

2.3.3 Agente deliberativo: orientado por objectivos

Os agentes orientados por objectivos utilizam objectivos para planear e escolher a acção. Este tipo de agentes é semelhante ao agente reactivo com estado onde a utilização de objectivos permite uma maior flexibilidade, pois para o mesmo estado o agente pode ter comportamentos diferentes.

2.3.4 Agente deliberativo: orientado por função de utilidade

A função utilidade permite medir numericamente a satisfação do agente, ao executar uma determinada acção. Os agente orientados por funções de utilidade determinam as acções a tomar, escolhendo a acção que tenha o valor de utilidade mais elevado.

2.4 Estudo de arquitecturas de equipas da MSL

2.4.1 Tribots

A equipa Tribots[2] resulta do grupo de investigação de Neuro-Informática da Universidade Alemã de Osnabrück.

A sua arquitectura [14–16] é baseada em comportamentos, herdando alguns conceitos de arquitecturas *Belief-Desire-Intention* (BDI) [17]. Neste arquitectura cada comportamento tem uma interface que verifica as condições de activação e uma interface que verifica se as condições de satisfação estão cumpridas. O comportamento está activo enquanto a condição de satisfação não está satisfeita.

Um árbitro, usando o conceito BDI, é utilizado para gerir os comportamentos de um determinado tipo de agente. Contextualizando esta arquitectura nos conceitos BDI, a crença (*belief*) é o estado do mundo, os desejos (*desire*) são os comandos enviados aos actuadores e a intenção (*intention*) é o comportamento activo. Os comportamentos estão ordenados do mais prioritário para o menos prioritário. Utilizando diferentes tipos de árbitros é possível escolher os comportamentos segundo as suas prioridades ou executar um comportamento até que se verificar a condição de satisfação. Estes árbitros BDI são eles próprios um comportamento, assim sendo, é possível que sejam utilizados por outros comportamentos, criando uma hierarquia de comportamentos.

Esta abordagem pode ser comparada com uma máquina de estados finitos, onde as transições não são especificadas (ver figura 2.11). Este facto permite que sejam adicionados e retirados estados sem originar problemas na definição de transições.

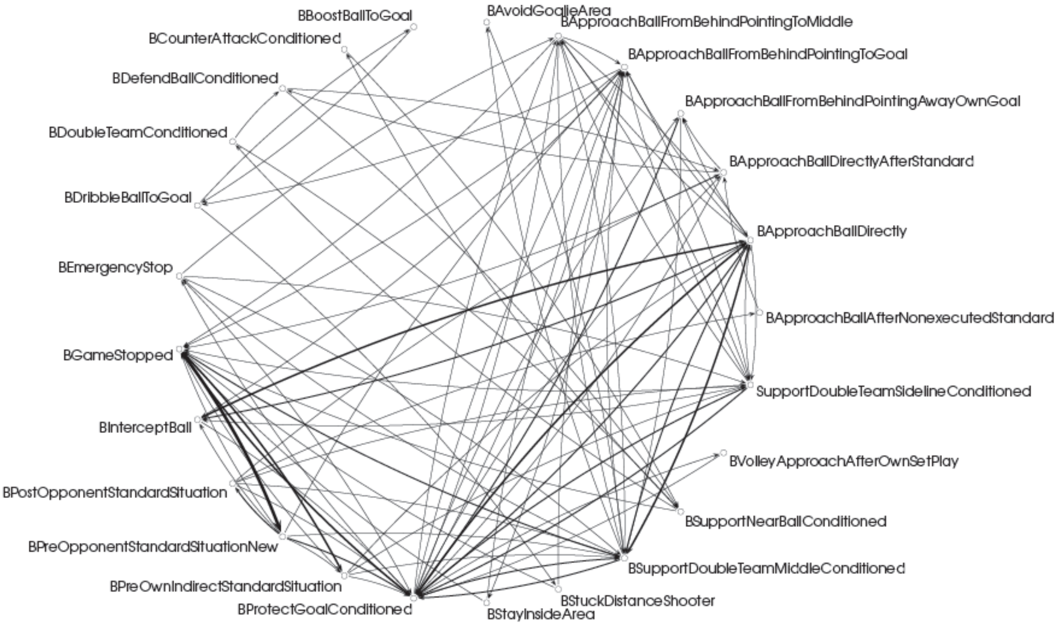

BBoostBallToGoal
BAvoidGoalieArea
BApproachBallFromBehindPointingToMiddle
BCounterAttackConditioned
BApproachBallFromBehindPointingToGoal
BDefendBallConditioned
BApproachBallFromBehindPointingAwayOwnGoal
BDoubleTeamConditioned
BApproachBallDirectlyAfterStandard
BDribbleBallToGoal
BApproachBallDirectly
BEmergencyStop
BApproachBallAfterNonexecutedStandard
BGameStopped
SupportDoubleTeamSidelineConditioned
BInterceptBall
BVolleyApproachAfterOwnSetPlay
BPostOpponentStandardSituation
BSupportNearBallConditioned
BPreOpponentStandardSituationNew
BSupportDoubleTeamMiddleConditioned
BPreOwnIndirectStandardSituation
BStuckDistanceShooter
BProtectGoalConditioned
BStayInsideArea

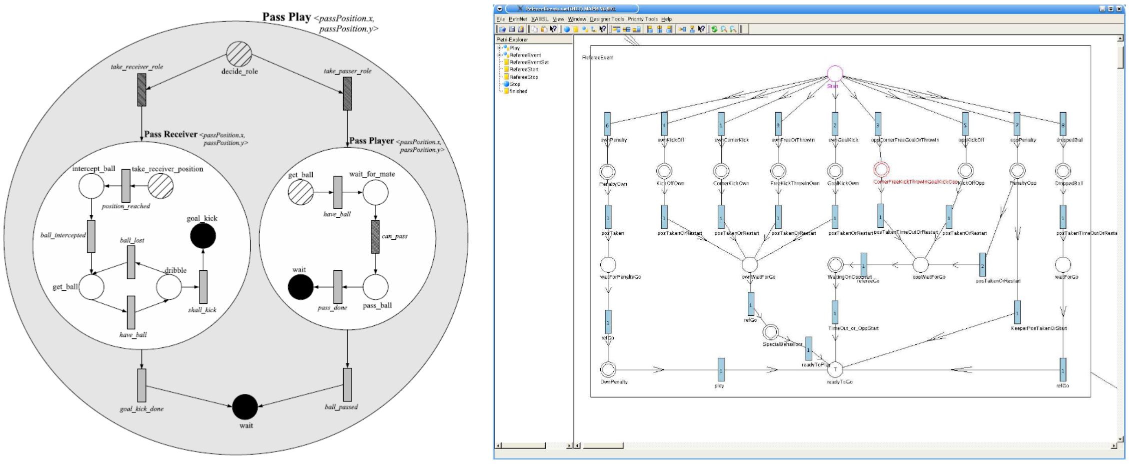
Figura 2.11: Diagrama de comportamentos equipa Tribots

---

[2] `http://www.ni.uos.de/index.php?id=25`

2.4.2 COPS

A equipa COPS (*Cooperative sOccer Playing robotS*) da Universidade de Estugarda utiliza papéis para coordenar a sua equipa. Estes papéis são fixos durante o jogo, apesar de poderem ser alterados entre agentes de uma forma dinâmica. No caso de se definir papéis do mesmo tipo, existe a possibilidade de utilização de sub-papéis.


Pass Play <passPosition.x, passPosition.y>
decide_role
take_receiver_role
take_passer_role
Pass Receiver <passPosition.x, passPosition.y>
Pass Player <passPosition.x, passPosition.y>
intercept_ball
take_receiver_position
position_reached
ball_intercepted
goal_kick
ball_lost
dribble
get_ball
shall_kick
have_ball
get_ball
wait_for_mate
have_ball
can_pass
wait
pass_done
pass_ball
goal_kick_done
ball_passed
wait

(a) Rede de passe em jogo

(b) Editor de redes XPlM

Figura 2.12: Dois exemplos de redes XPlM

Para modelar a coordenação entre agentes são utilizadas redes de decisão, mais concretamente, redes XPlM [18, 19] que são um sub-conjunto de redes de Interacção. A nível dos comportamentos, é utilizada a linguagem de especificação de comportamentos XABSL [20]. A figura 2.12 ilustra o editor desta linguagem.

2.5 Sumário

Neste capítulo foram explicados alguns conceitos necessários para a contextualização desta dissertação, nomeadamente, o conceito de agente. De seguida foram apresentados os conceitos dos vários tipos de agentes. Por fim foi elaborado um estudo das arquitecturas utilizadas por duas equipas: os Tribots da Universidade de Osnabrück e os COPS da Universidade de Estugarda.

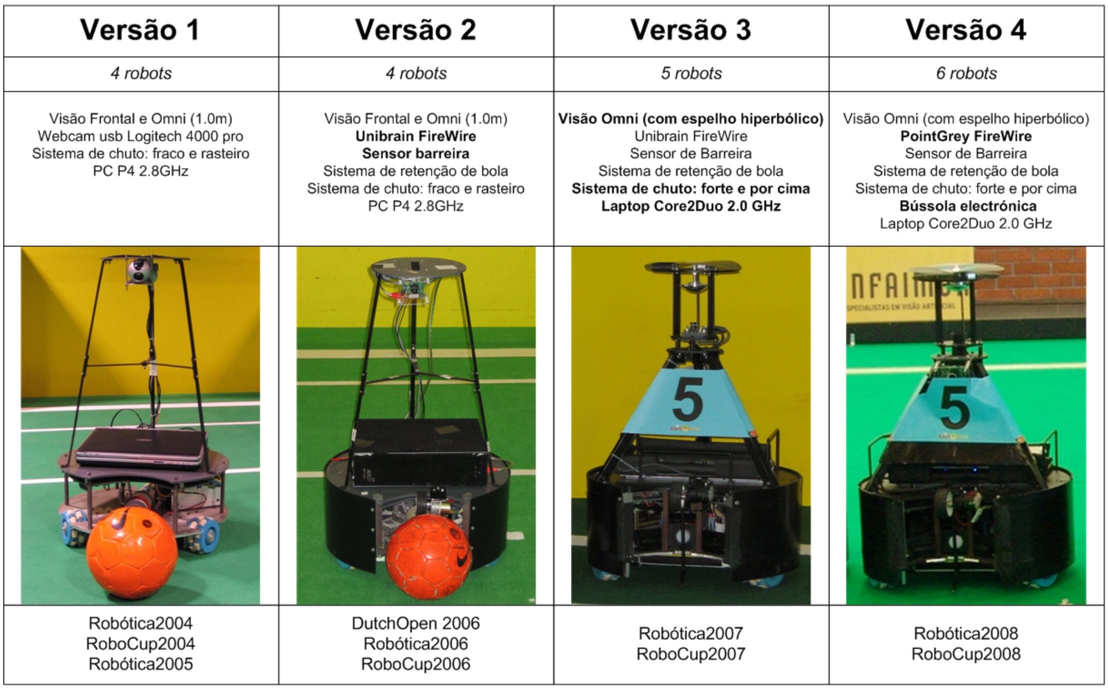
Capítulo 3

O robô CAMBADA

No capítulo 2 secção 2.2 foram apresentadas soluções a nível de estrutura física, sensores e actuadores, tendo sido algumas delas implementadas pela equipa CAMBADA.

A equipa CAMBADA é uma equipa constituída por seis robôs completamente autónomos. Este conjunto de robôs tem como principal objectivo jogar futebol na Liga dos Médios, *Middle Size League* (MSL) do RoboCup.

Na figura 3.1 pode ver-se a evolução dos robôs da equipa CAMBADA. Como suporte à implementação e validação do trabalho desenvolvido foram usadas as versões 2 e 3/4.



Versão 1	Versão 2	Versão 3	Versão 4
4 robots	*4 robots*	*5 robots*	*6 robots*
Visão Frontal e Omni (1.0m) Webcam usb Logitech 4000 pro Sistema de chuto: fraco e rasteiro PC P4 2.8GHz	Visão Frontal e Omni (1.0m) **Unibrain FireWire** **Sensor barreira** Sistema de retenção de bola Sistema de chuto: fraco e rasteiro PC P4 2.8GHz	**Visão Omni (com espelho hiperbólico)** Unibrain FireWire Sensor de Barreira Sistema de retenção de bola **Sistema de chuto: forte e por cima** **Laptop Core2Duo 2.0 GHz**	Visão Omni (com espelho hiperbólico) **PointGrey FireWire** Sensor de Barreira Sistema de retenção de bola Sistema de chuto: forte e por cima **Bússola electrónica** Laptop Core2Duo 2.0 GHz
Robótica2004 RoboCup2004 Robótica2005	DutchOpen 2006 Robótica2006 RoboCup2006	Robótica2007 RoboCup2007	Robótica2008 RoboCup2008

Figura 3.1: Evolução dos robôs CAMBADA

O robô CAMBADA tem uma altura de 76 cm e uma base circular, com 48cm de diâmetro. O robô CAMBADA pode separar-se em três partes: a base, o sistema computacional e o sistema de visão.

3.1 Base

Como já foi referido anteriormente, a base do robô CAMBADA é circular, com cerca de 48 cm de diâmetro, composta por actuadores, tais como: motores, sistema de chuto e sistema de retenção de bola. Este base também dispõe de um sensor de barreira (ver figura 2.5). Para controlar os actuadores e sensores existe uma camada de electrónica designada por *hardware* de controlo.

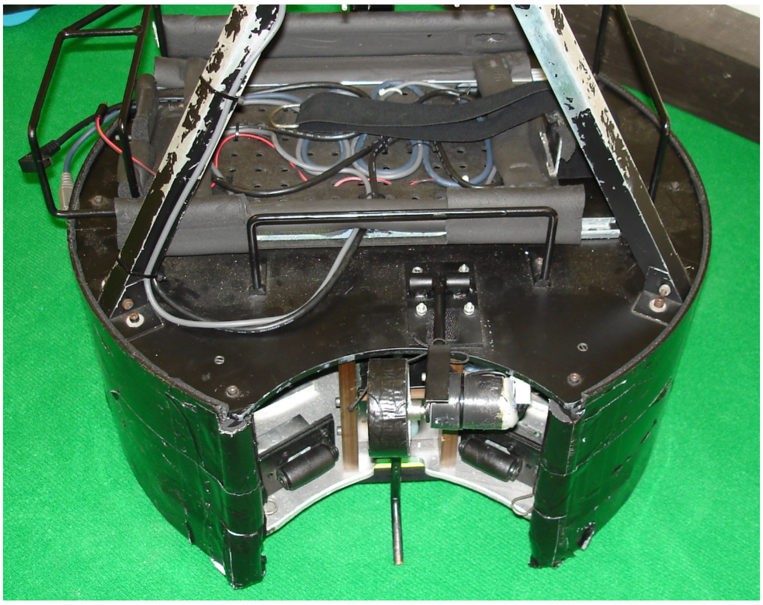


Figura 3.2: Base

Actuadores

Para a locomoção o robô CAMBADA utiliza três de motores Maxon [21], de 150W e 24V e três rodas omni direccionais. Este conjunto permite um movimento holonómico, ou seja, movimento em qualquer direcção e com qualquer orientação.

O sistema de retenção de bola é composto por um motor vulgar de 12V e uma roda usada em carros de modelismo. Este sistema permite controlar a bola a quando a condução.

O sistema de chuto é electromagnético, baseado no princípio de uma *coil Gun* [22, 23].

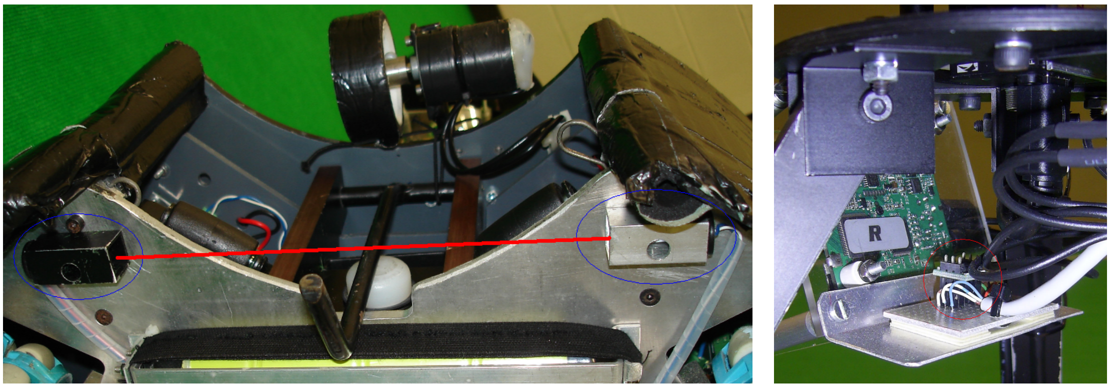
Para suprir as necessidades energéticas, são usadas três baterias 12V de Níquel Metal-Híbrido (NiMh). Este conjunto de baterias permite uma autonomia de duas horas.

Sensores

O sensor de barreira (ver figura 2.5) permite detectar se bola está encaixada na frente do robô. Esta informação é de elevada importância pois permite saber se a bola é afectada pelo sistema de chuto. Este sensor é composto por um receptor e emissor de infra-vermelhos. Este conjunto de receptor/emissor criam entre eles um feixe de infra-vermelho. Na situação em que a bola está encaixada na frente do robô, interrompe esse feixe.

O deslocamento das rodas do robô é medido através de sensores específicos, localizados nos motores. Estes sensores são designados por *encoders*. A conjugação desta informação permite gerar informação de odometria que posteriormente enviada para o agente.

A bússola electrónica permite que se tenha uma informação da orientação absoluta. Esta informação permite que se detecte situações de erro na auto-localização, através da comparação da orientação calculada pelo agente e a orientação efectiva.



(a) barreira (b) bussola

Figura 3.3: Sensor barreira e bússola electrónica

3.1.1 *Hardware*

A arquitectura de *hardware* é baseada numa arquitectura distribuída [24]. O *hardware* é composto por um conjunto de micro-controladores [25] e por electrónica de ligação. Estes micro-controladores estão interligados por uma rede CAN [26] [27], a qual utiliza o protocolo *Flexible Time-Triggered communication over CAN* (FTT-CAN) [28], de modo a permitir uma uma sincronização entre os módulos e cumprir os requisitos de tempo real.

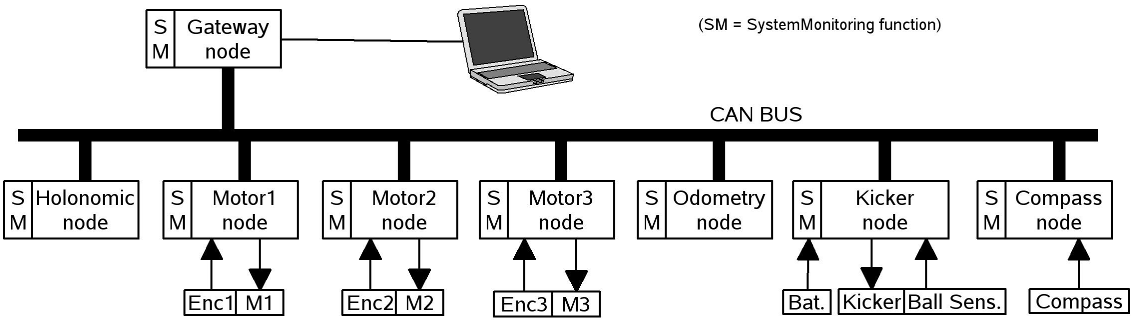

S M Gateway node
(SM = SystemMonitoring function)
CAN BUS
S M Holonomic node
S M Motor1 node
S M Motor2 node
S M Motor3 node
S M Odometry node
S M Kicker node
S M Compass node
Enc1
M1
Enc2
M2
Enc3
M3
Bat.
Kicker
Ball Sens.
Compass

Figura 3.4: *Hardware - lógica de controlo*

3.1.2 Sistema de visão

O principal sensor é a visão, este permite detectar os principais objectos do campo de futebol robótico. Os objectos detectados são: a bola, as linhas do campo e os obstáculos. O sistema de visão foi reformulado da versão 2 para a versão 3/4.

O sistema de visão da versão 2 era composto por duas câmaras *FireWire* Unibrain[1]. A câmara frontal permitia a detecção da baliza a 12m, a bola a 6m e os obstáculos a 3m. A outra câmara era omnidireccional de curto alcance e permitia detectar a bola a menos de 1m.

A versão 3/4 dispõe de um sistema de visão catadióptrico composto por um espelho hiperbólico e uma câmara *FireWire*[2] PointGrey[3].

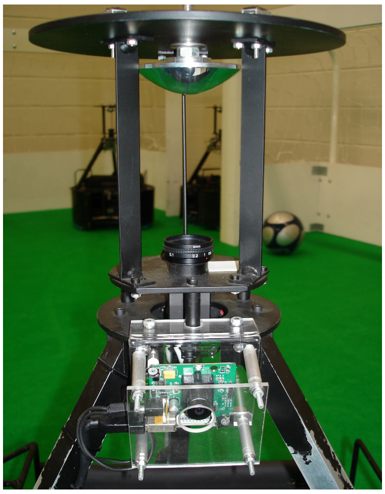


Figura 3.5: *Sistema de visão*

3.1.3 Sistema Computacional

Na versão 2, o sistema computacional era um PC vulgar de secretária, com um processador Intel Pentium 4 a 2.8GHz e com 512MB de memória RAM. Este utilizava como unidade de armazenamento uma *Compact Flash* de 512MB. O sistema operativo escolhido é GNU/Linux [29], tendo sido feita uma instalação *from scratch*, através do *debootstrap* da distribuição *Debian* [30]. Este sistema dispunha de uma autonomia inferior a 45 minutos.

O sistema computacional utilizado actualmente na versão 3/4 é um computador portátil Fujitsu-Siemens 12", com um processador Intel Core2Duo [31], 1GB de memória RAM e dispõe de uma autonomia de duas horas. O sistema operativo escolhido é o GNU/Linux [29], mais especificamente a distribuição Ubuntu [32]. Este sistema computacional, devido ao seu processador Core2Duo, permite uma execução paralela de tarefas.

3.2 Sumário

Neste capítulo efectuou-se um descrição das principais características da estrutura que suporta o robô CAMBADA, na sua versão 2 e 3/4. Assim, apresentaram-se as principais características dos actuado-

---

[1] `http://www.unibrain.com/Products/VisionImg/Fire_i_BC.htm`

[2] `http://pt.wikipedia.org/wiki/FireWire`

[3] `http://www.ptgrey.com`

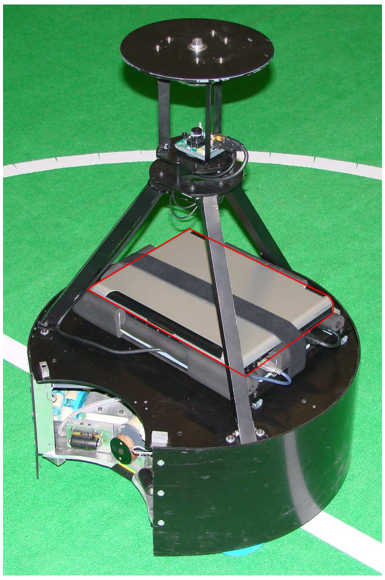


Figura 3.6: Sistema computacional

res, sensores, *hardware*, sistema de visão e sistema computacional do robô CAMBADA.

Capítulo 4

Arquitectura de *Software* da equipa CAMBADA

No capítulo anterior foram apresentados os sensores e actuadores que estão presentes nos robôs CAMBADA.

O *software* da equipa CAMBADA divide-se em três componentes: arquitectura de *software*, arquitectura do agente e gestão do estado do mundo. A arquitectura de *software* permite ter uma visão geral dos vários módulos de *software* das diversas áreas e as relações entre eles. A arquitectura do agente permite demonstrar como são interpretadas as percepções recolhidas pelos sensores e de que modo são geradas as acções que são enviadas para os actuadores. A gestão do estado do mundo permite a organização, filtragem e agregação da informação recolhida pelos sensores, de modo a facilitar o acesso por parte do agente.

4.1 Arquitectura de *software*

A arquitectura de *software* é composta por seis componentes: RtDb[1], Comm, HWComm, Monitor, Vision e CambadaAgent, como ilustra a figura 4.1.

A **RtDb** [33] é o componente desta arquitectura de *software* que permite armazenar e fornecer mecanismos para o acesso à informação persistente. Cabe, também, a este componente providenciar um canal de comunicação para a troca de informação entre os vários processos deste sistema. Através da figura 4.1 visualiza-se que **RtDb** é constituída por duas partes, uma partilhada e outra local. Actualmente, a **RtDb** é implementada através da memória partilhada, fornecida pelo sistema operativo *GNU/Linux* [29, 32].

O **Comm** é o componente [34–36] responsável pela troca/difusão da informação entre os vários agentes. A informação partilhada por todos os agentes CAMBADA é a informação que está localizada na parte partilhada na **RtDb**. Este componente usa o protocolo *Multicast* para a difusão da informação.

O **HWComm** é responsável pela comunicação entre a unidade computacional (PC portátil) e a base. No sentido PC-BASE fluem as mensagens que permitem controlar os motores, o sistemas de retenção

---

[1] **R**eal**T**ime **D**ata**B**ase

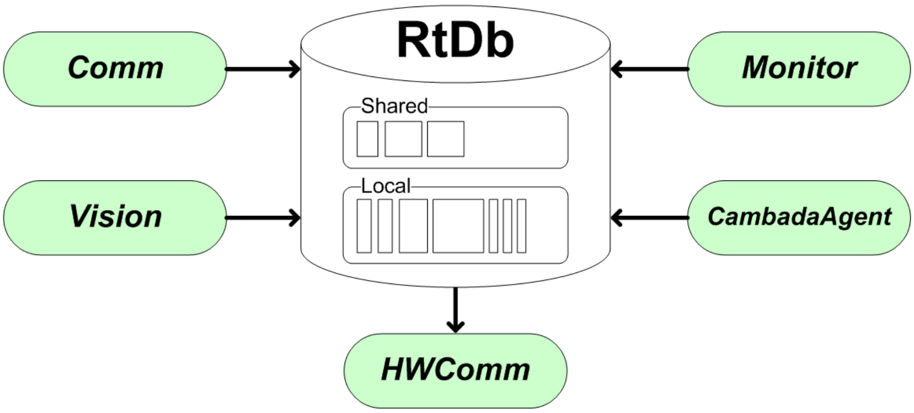

RtDb
Shared
Local
Comm
Vision
Monitor
CambadaAgent
HWComm

Figura 4.1: Arquitectura de *software*

de bola e o sistema de chuto. Por outro lado, no sentido BASE-PC fluem as mensagens de monitorização da tensão das baterias, o estado do sensor de barreira e informação de odometria.

Com o trabalho de desenvolvimento do componente **Monitor** pretendeu-se suprimir duas lacunas do sistema, designadamente, saber o estado dos processos principais que permitem o funcionamento do robô CAMBADA e a monitorizar a bateria da unidade computacional.

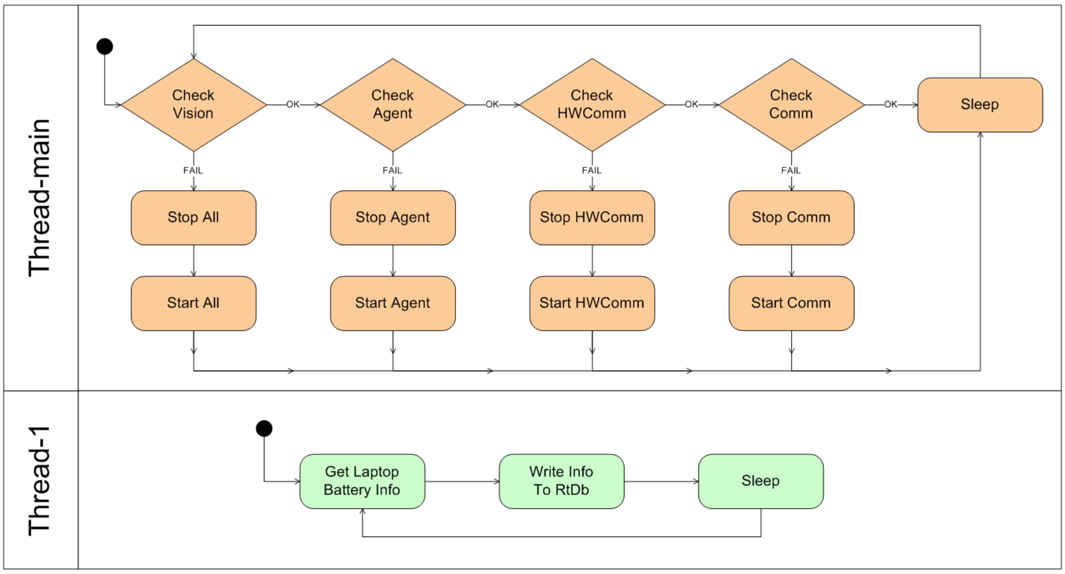

Thread-main
Check Vision
OK
Check Agent
OK
Check HWComm
OK
Check Comm
OK
Sleep
FAIL
Stop All
Stop Agent
Stop HWComm
Stop Comm
Start All
Start Agent
Start HWComm
Start Comm
Thread-1
Get Laptop Battery Info
Write Info To RtDb
Sleep

Figura 4.2: Diagrama de actividade do *Monitor*

Como se pode visualizar na figura 4.2 o **Monitor** é composto por dois fios de execução distintos. O fio de execução *Thread-main* permite monitorizar os processos Vision, HWComm, Comm e CambadaAgent. Esta monitorização permitir saber se os referidos processos estão activos, ou seja, se estão a correr e caso estejam a correr, se têm um funcionamento normal. Este funcionamento é inferido através do tempo de actualização da informação que os processos produzem. No caso de um processo não estar a funcionar correctamente, a decisão tomada é parar e arrancar novamente o processo.

O fio de execução *Thread-1* é responsável pela monitorização da percentagem de carga da bateria e do seu estado. Este estado tem três valores possíveis: a descarregar, a carregar e completamente carregado.

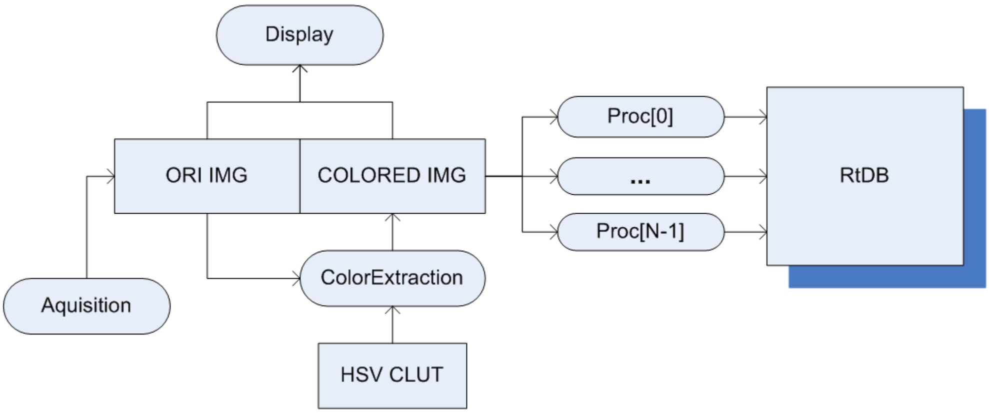
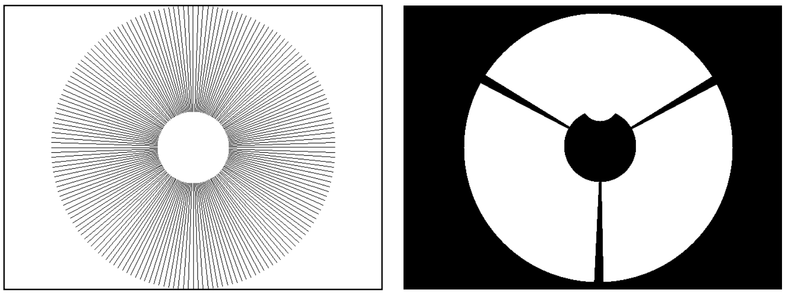
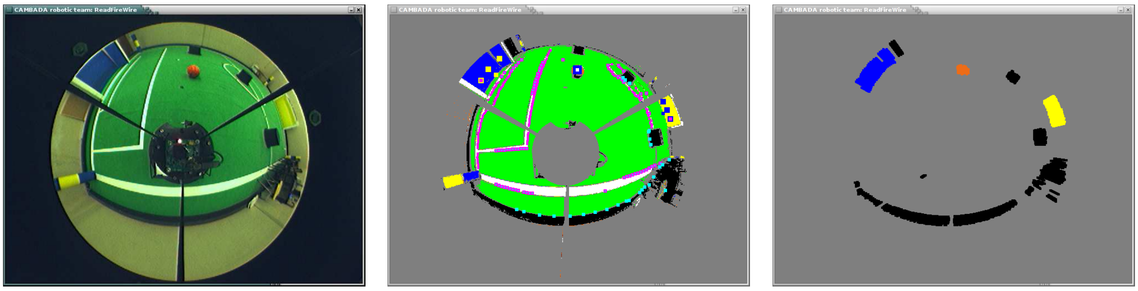


Figura 4.3: Arquitectura do componente *Vision*

No componente **Vision** o trabalho centrou-se na definição da arquitectura. Este componente é responsável pela aquisição e processamento de imagem. O processamento de imagem tem como objectivo a detecção de vários elementos que estão presentes no campo de futebol robótico. Os principais objectos a serem detectados são a bola (fifa nº5 cor de laranja), os obstáculos e as linhas brancas do campo [37].



(a) Sensores radiais (b) Máscara



(c) Imagem original (d) Imagem segmentada (e) Imagem processada

Figura 4.4: Sistema de visão

A figura 4.4(a) permite visualizar os sensores radiais [38,39] usados para a detecção dos objectos. Os sensores radiais são previamente criados segundo o algoritmo de Bresenham [40,41]. O uso deste tipo de sensores permite numa imagem de 640x480 pixeis, uma redução de 50%. Ainda é aplicada uma máscara (ver figura 4.4(b)) para não se processar os pixeis que estão no corpo do robô, o que permite uma nova redução de 50% no número de pixeis processados.

O componente **CambadaAgent** é responsável pelo processamento das percepções e geração de ordens para os actuadores. Este componente vai ser descrito detalhadamente na secção 4.2.

A sincronização dos processos envolvidos nesta arquitectura é garantida através da biblioteca *Pman* [**?**] (Process Manager).

4.2 Arquitectura do agente CAMBADA

A missão de equipa de robôs futebolistas é muito complexa. Pelo facto do seu objectivo ser complexo o de jogar futebol, e também ao pelo facto do ambiente de actuação ser extremamente dinâmico.

A necessidade de especificar e implementar uma arquitectura de *software*, resulta da complexidade das tarefas que uma equipa de robôs futebolistas se propõe resolver.

A arquitectura que foi especificada e implementada baseia-se em três pontos fundamentais: a gestão de informação, comportamentos básicos [42] e papéis. A gestão de informação usada neste arquitectura vai ser analisada em detalhe na secção 4.3.

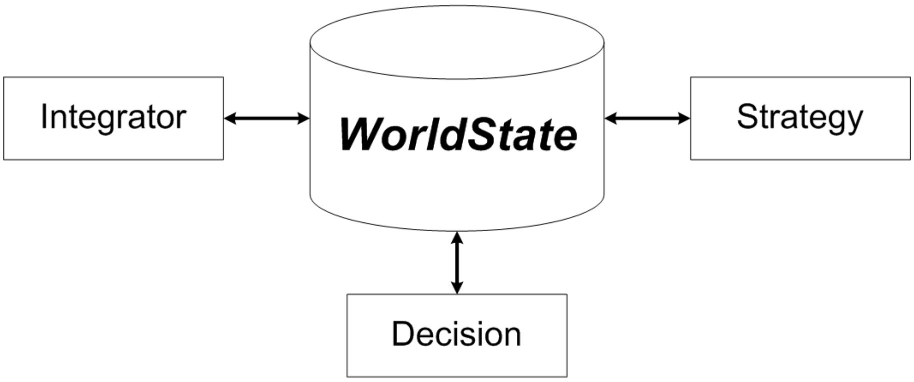


Figura 4.5: Arquitectura do agente CAMBADA

Como se comprova através da figura 4.5 é possível separar a arquitectura implementada em quatro partes, o estado do mundo, *Integrator*, *Strategy* e o *Decision*.

4.2.1 *Integrator*

O *Integrator* é o módulo responsável por converter a informação recolhida pelos sensores em informação tratada e usável. Este módulo é igualmente responsável por integrar as informações difundidas por outros agentes e pela *BaseStation* [43]. A *BaseStation* é o *software* responsável pelo processamento das ordens enviadas pela *Referee-Box*, pelo controlo e monitorização do estado interno dos robôs CAMBADA. O *Coach* vai ser detalhado no capítulo 5, por ser usado na coordenação da equipa.

O *Integrator* começa por integrar a informação que vem da Base, tensão das baterias e odometria. De seguida, integra a informação proveniente da *BaseStation*, cor da equipa, baliza a atacar, estado do robô (*running*), o seu papel e o *game-state*. O motivo de se poder definir o papel através da *BaseStation* prende-se com a possibilidade de se efectuarem testes a um determinado papel, durante a competição a informação enviada é *Auto*, possibilitando aos agentes determinarem o melhor papel para si. Continuando a análise, constata-se que este módulo efectua a integração de informação dos colegas de equipa. Esta integração é feita, para recolher a informação do papel, cor de equipa, estado do robô e a baliza a atacar. Caso um robô deixe de funcionar é necessário corrigir o seu estado e passá-lo para *notRunning*. Esta detecção é feita através da *RtDb* e a sua funcionalidade de permitir saber à quando tempo foi actualizada a informação. Depois da integração dos obstáculos e da bússola, é feita a integração da localização, a qual é feita com a informação da detecção das linhas do campo [37], a posição anterior e o módulo de localização [44]. De seguida é realizada a certificação da localização, com base na bússola. Essa detecção, como ilustra a figura 4.6, é baseada na diferença entre a orientação dada pelo módulo de localização e pela orientação dada pela bússola. A figura 4.6 ilustra a certificação da localização e a classificação da diferença dos ângulos. A zona verde representa uma localização correcta, a zona amarela representa uma localização correcta mas invertida. Para a corrigir esta situação é necessário apenas fazer *mirror* da localização. As zonas vermelhas representam uma localização errada, logo é necessário forçar uma relocalização.

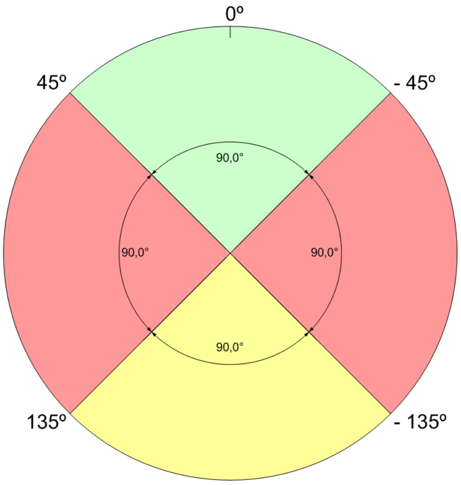


Figura 4.6: Detecção de erros de localização com base na bússola electrónica

O *Integrator* integra a informação da posição da bola da visão omni, da visão frontal e dos outros agentes. Cabe a este módulo a tarefa de integrar sucessivas posições da bola, determinar a sua velocidade e filtrar o ruído existente [45], para esse efeito é usado um filtro de Kalman [46].

No *gameState* a integração feita é no sentido de criar estados a partir dos sinais vindos da *RefereeBox*. A figura 4.7 ilustra o diagrama de estados do *gameState*. Cada falta dá origem a dois estados, o *pre* e o *post*. No *pre* os robôs colocam-se em posição. No *post*, se as faltas forem a favor, os robôs executam as jogadas estudadas, no caso de serem contra, vão esperar que passem os dez segundos ou que a bola se mova.

4.2.2 *Strategy*

O módulo *Strategy* é responsável pela gestão da estratégia da equipa CAMBADA e vai ser detalhado no capítulo 5.

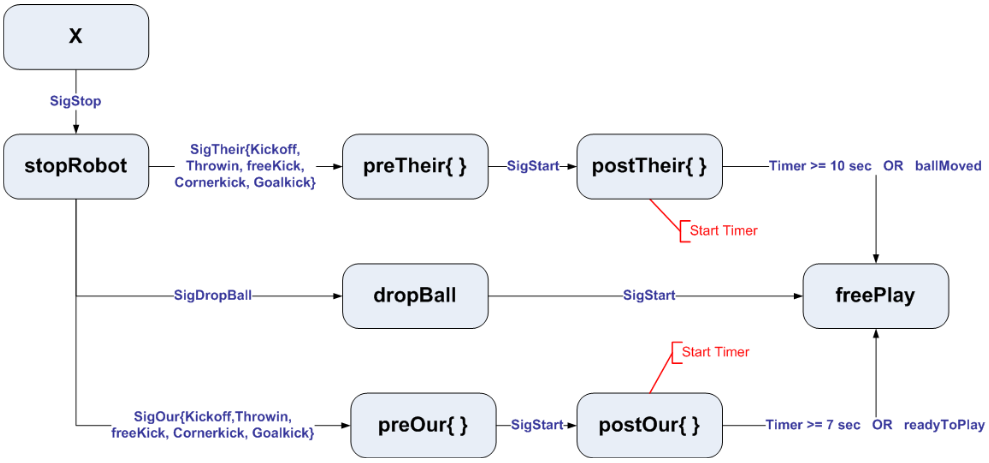

X
SigStop
stopRobot
SigTheir{Kickoff, Throwin, freeKick, Cornerkick, Goalkick}
preTheir{ }
SigStart
postTheir{ }
Timer >= 10 sec OR ballMoved
Start Timer
SigDropBall
dropBall
SigStart
freePlay
Start Timer
SigOur{Kickoff,Throwin, freeKick, Cornerkick, Goalkick}
preOur{ }
SigStart
postOur{ }
Timer >= 7 sec OR readyToPlay

Figura 4.7: Diagrama de estados do *GameState*

4.2.3 *Decision*

Decision é o módulo que implementa a decisão das ordens a enviar para os actuadores, baseando-se em papéis e comportamentos.

Este módulo tem um método chamado *roleSelection*, este escolhe o papel que o agente vai ter. O método de selecção dos papéis vai ser descrito no capítulo 5, o qual explica a coordenação da equipa CAMBADA. Depois da selecção do papel cabe ao módulo *Decision* activar este papel.

Papéis

O papéis são implementados através de uma classe abstracta *Role*. Na figura 4.8 pode ver-se que a classe *Role* apresenta dois métodos importantes: *run* e *changeBehaviour*. O método *run* é chamado para activar o papel. Esta activação é feita através da chamada do método abstracto *determineNextState* que dependendo de papel para papel tem uma implementação própria. Essa implementação permite determinar um comportamento cujo cálculo e execução é efectuado nesta fase. Importa salientar que para a implementação de um novo papel apenas é necessário implementar o método *determineNextState* da nova classe, do novo papel. O *changeBehaviour* é o método utilizado pelos papéis para usar/activar um comportamento.

Importa referir que a cada ciclo o papel não é destruído e nem inicializado novamente, o que significa que, possibilita a persistência de informação.

Comportamentos

Os comportamentos básicos são a base da decisão da equipa CAMBADA. Depois de se escolher o comportamento básico através de um papel, cabe ao comportamento converter uma tarefa em ordens para os actuadores.

Analisando a figura 4.9, pode-se constatar que a classe *Behaviour* tem seis métodos importantes: *reset*, *calculate*, *execute*, *grabberControl*, *kickTo* e *passTo*. Os métodos *reset* e *calculate* são abstractos e a sua implementação numa nova classe origina um novo comportamento. O *reset* é chamado quando há

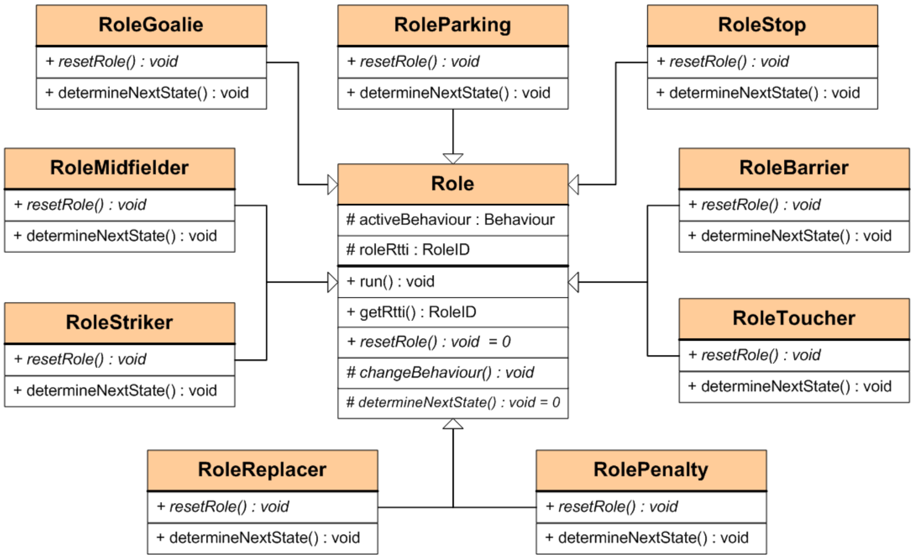

RoleGoalie
+ resetRole() : void
+ determineNextState() : void
RoleParking
+ resetRole() : void
+ determineNextState() : void
RoleStop
+ resetRole() : void
+ determineNextState() : void
RoleMidfielder
+ resetRole() : void
+ determineNextState() : void
Role
# activeBehaviour : Behaviour
# roleRtti : RoleID
+ run() : void
+ getRtti() : RoleID
+ resetRole() : void = 0
# changeBehaviour() : void
# determineNextState() : void = 0
RoleBarrier
+ resetRole() : void
+ determineNextState() : void
RoleStriker
+ resetRole() : void
+ determineNextState() : void
RoleToucher
+ resetRole() : void
+ determineNextState() : void
RoleReplacer
+ resetRole() : void
+ determineNextState() : void
RolePenalty
+ resetRole() : void
+ determineNextState() : void

Figura 4.8: Diagrama de Papéis

uma troca de comportamento, ou seja, quando o papel executou um *changeBehaviour* para o comportamento com um novo identificador. O *calculate* é o método que vai conter o código de selecção das ordens a enviar para os actuadores. Os métodos *kickTo* e *passTo* são usados para auxiliar na conversão de unidades SI em unidades do actuador.

O método *grabberControl* permite o controlo automática do sistema de retenção de bola. Este sistema activa-se automaticamente quando a bola se encontra dentro de um arco com 90 graus de abertura e 1.5 metros de raio, como ilustra a figura 4.10.

4.3 Gestão da informação do agente CAMBADA

Para garantir uma fácil avaliação do estado do mundo e que a melhor decisão seja tomada foi necessário criar uma estrutura de suporte à informação recolhida. Assim sendo, foi criado o conceito de *WorldState* cuja organização é reflectida na figura 4.11.

A classe *ZoneMatrix* permite a descretização do campo em células de 50cm. Cada célula pode ser do tipo: começo, destino, obstáculo e livre. Esta classe permite determinar o caminho mais curto usando o algoritmo A* [47]. Este algoritmo fornece o caminho mais curto entre a célula de começo e a célula de destino. Este algoritmo foi modificado de modo a não considerar células a uma distância *n* de células com obstáculo. A figura 4.12 ilustra dois caminhos calculados, a célula de começo (azul) e a célula de destino (vermelho). O caminho representado pela cruz magenta é o caminho calculado pelo A* e o amarelo representa o caminho calculado pelo o A* modificado. Esta modificação deve-se ao facto do caminho pretendido evitar os adversários, ou seja, evitá-los não apenas por causa das colisões, mas para manter uma distância segura para que eles não roubarem a bola.

O *WorldState* é o repositório de informação tratada que vai ser usada pelo agente que selecciona as acções a executar. É providenciado por esta estrutura três grupos de método. O primeiro grupo é composto por métodos de transformação de sistema de coordenadas. Permite conversões do sistema de coordenadas relativas do robô em coordenadas absolutas, e o contrário. A figura 4.13 ilustra os

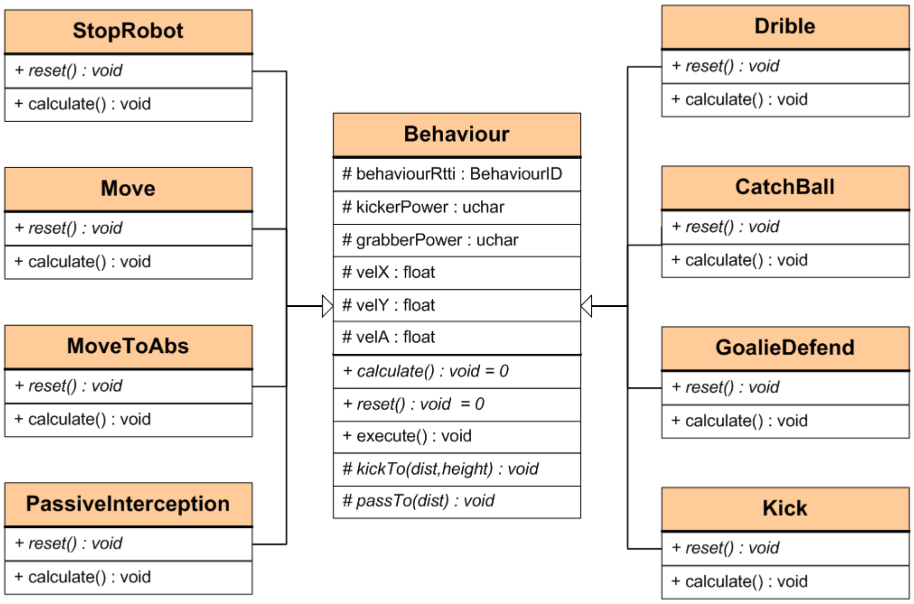

StopRobot
+ reset() : void
+ calculate() : void
Move
+ reset() : void
+ calculate() : void
MoveToAbs
+ reset() : void
+ calculate() : void
PassiveInterception
+ reset() : void
+ calculate() : void
Behaviour
# behaviourRtti : BehaviourID
# kickerPower : uchar
# grabberPower : uchar
# velX : float
# velY : float
# velA : float
+ calculate() : void = 0
+ reset() : void = 0
+ execute() : void
# kickTo(dist,height) : void
# passTo(dist) : void
Drible
+ reset() : void
+ calculate() : void
CatchBall
+ reset() : void
+ calculate() : void
GoalieDefend
+ reset() : void
+ calculate() : void
Kick
+ reset() : void
+ calculate() : void

Figura 4.9: Diagrama de Comportamentos

dois sistemas de coordenadas, os eixos vermelhos representam o sistema de coordenadas absolutas, sendo o sentido de jogo da esquerda para a direita, o sistema de coordenadas relativas ao robô são representados pelos eixos azuis. O segundo grupo é composto por métodos de perguntas ao estado do mundo. Neste grupo existem funções que retornam informações tais como o robô mais próximo da bola, os robôs que têm um determinado papel, ou se está um posição de remate, se está perto de uma determinada área. Também existem métodos que fornecem a melhor direcção para conduzir a bola, segundo um campo de utilidades ou segundo um mapa de ocupação. Estas perguntas permitem simplificar a análise das acções a tomar.

O terceiro e último grupo de métodos é um conjunto de métodos que fornecem as dimensões do campo, alguns pontos chave e as dimensões das áreas. Para se manter uma boa abstracção e separação entre as percepções e a centro de decisão é necessário.

4.4 Distribuição de código em sistemas distribuídos

Após, a especificação e implementação da arquitectura de *software*, controlo e coordenação dos agentes CAMBADA, existia a necessidade de difundir o *software* de controlo pelos vários agentes, visto que a equipa CAMBADA não possuía nenhum mecanismo de distribuição.

Para suprimir esta lacuna e finalizar o trabalho de especificação e implementação da arquitectura de *software* dos agentes CAMBADA foi desenvolvido o *sendToCambadas*. Este é um *script* implementado na linguagem de *scripting bash* [48], usando o *rsync* sincroniza o código e os ficheiros de configuração do PC de compilação para os agentes. Esta sincronização pode ser parcial, de modo a sincronizar apenas alguns agentes definidos pelo utilizador. Após a sincronização é enviado a cada agente um sinal de reconfiguração para estes lerem novamente os ficheiros de configuração.

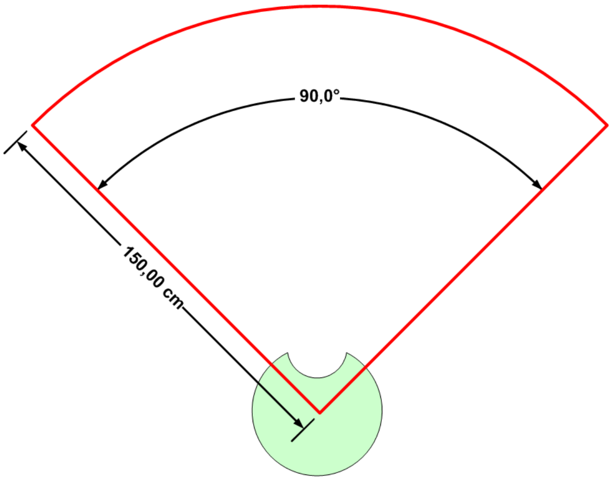

90,0°
150,00 cm

Figura 4.10: Zona de activação do dispositivo de retenção de bola

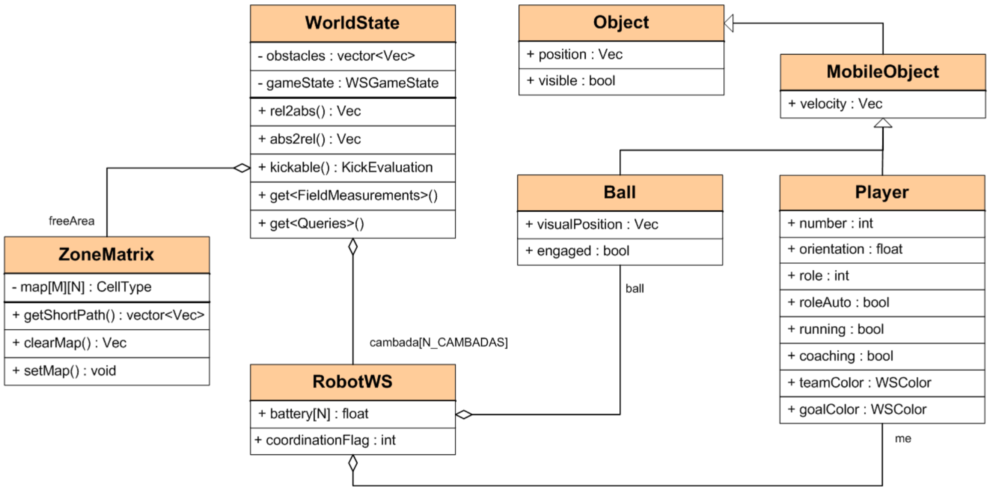

WorldState
- obstacles : vector<Vec>
- gameState : WSGameState
+ rel2abs() : Vec
+ abs2rel() : Vec
+ kickable() : KickEvaluation
+ get<FieldMeasurements>()
+ get<Queries>()
Object
+ position : Vec
+ visible : bool
MobileObject
+ velocity : Vec
Ball
+ visualPosition : Vec
+ engaged : bool
Player
+ number : int
+ orientation : float
+ role : int
+ roleAuto : bool
+ running : bool
+ coaching : bool
+ teamColor : WSColor
+ goalColor : WSColor
freeArea
ZoneMatrix
- map[M][N] : CellType
+ getShortPath() : vector<Vec>
+ clearMap() : Vec
+ setMap() : void
ball
cambada[N_CAMBADAS]
RobotWS
+ battery[N] : float
+ coordinationFlag : int
me

Figura 4.11: Diagrama de classes do estado do mundo

4.5 Sumário

Neste Capítulo foram descritos todos os componentes da arquitectura *software*. De seguida apresentou-se o trabalho desenvolvido na especificação e desenvolvimento da arquitectura de *software* da equipa CAMBADA, com especial atenção na arquitectura do agente e seus mecanismos de gestão da informação.

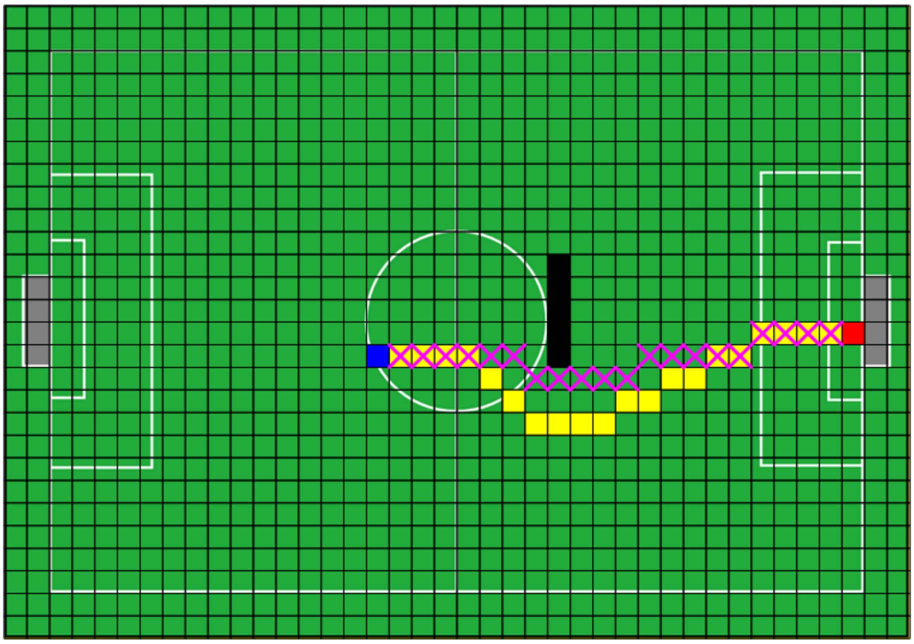


Figura 4.12: Caminhos calculados por A* e A* modificado

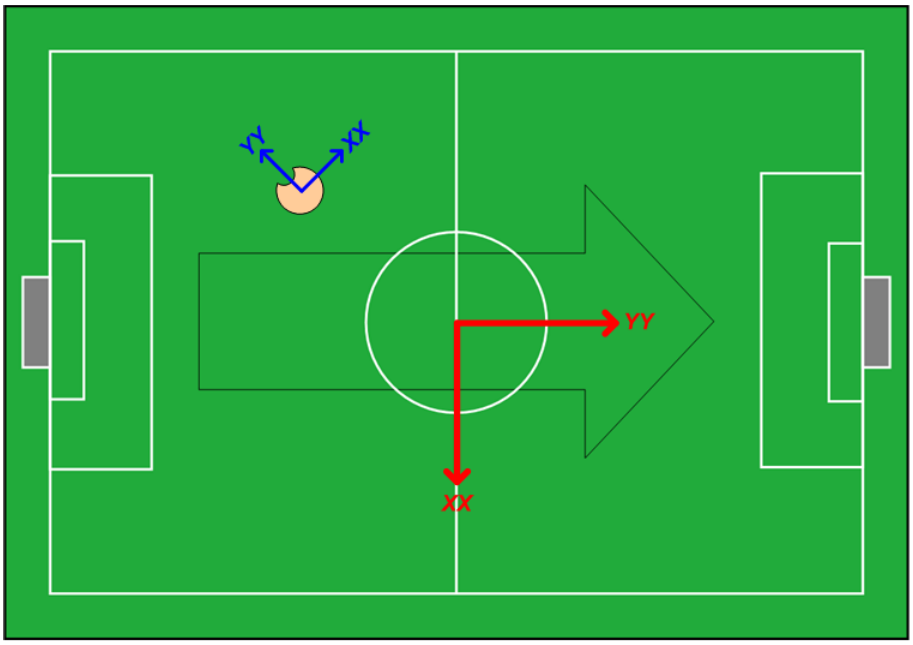


Figura 4.13: Sistemas de coordenadas (absoluto e relativo) utilizado pelo agente CAMBADA

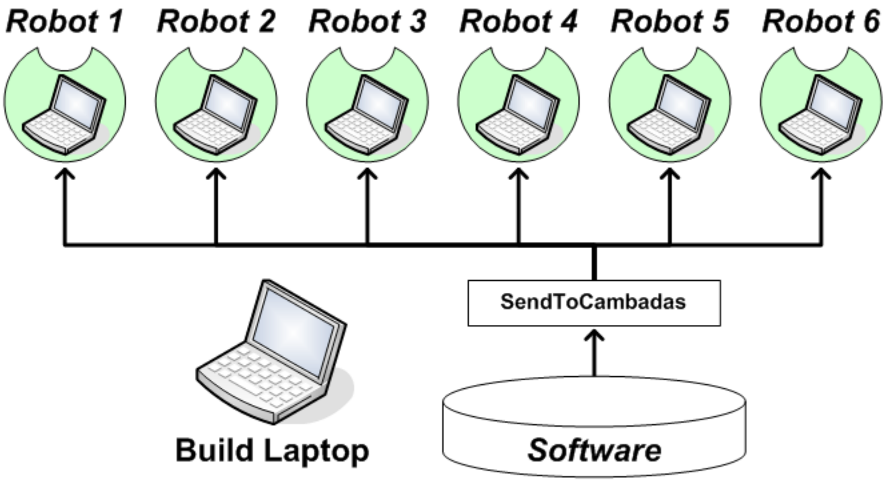

Robot 1
Robot 2
Robot 3
Robot 4
Robot 5
Robot 6
SendToCambadas
Build Laptop
Software

Figura 4.14: Exemplo de aplicação do *sendToCambadas*

Capítulo 5

Coordenação da equipa CAMBADA

Após a descrição do trabalho desenvolvido na arquitectura de *sofware*, controlo e coordenação da equipa CAMBADA, segue-se a descrição/comparação dos mecanismos de coordenação usados em dois momentos distintos, na versão 2 e na versão 3/4. É de salientar que as versões 2 e 3/4 usam a arquitectura de *software* descrita no capítulo anterior.

Os mecanismos de coordenação são necessários para minimizar o esforço de cada agente, bem como optimizar os recurso da equipa. Nas versões 2 e 3/4 são usados papéis para coordenar os vários agentes CAMBADA, onde cada agente tem um determinado papel atribuído. Esta atribuição de papéis é feita de um modo coordenado e dinâmico onde cada agente sabe o seu papel em campo e o papel dos colegas de equipa.

5.1 Versão 2

Como foi descrito no capítulo 3, a versão 2 apenas apresentava uma visão frontal e uma visão omnidireccional de curto alcance (50 cm), não dispunha de um módulo de auto-localização. O sistema de chuto apenas chutava bolas rasteiras e sem grande força. O primeiro ponto de trabalho foi a análise de requisitos de coordenação, tendo em conta que a informação disponível era apenas relativa.

Cada papel é uma composição de comportamentos básicos. Nesta versão cada papel é implementado segundo uma máquina de estados. Nesta versão do robô foram desenvolvidos para o agente CAMBADA quatro papéis, como ilustra a figura 5.1. Os papéis *Striker*, *Defender* e *Goalie* são usados em jogo. Em situações de reposição de bola do adversário é usado o papel *Midfielder*. O papel *Test* é usado apenas para efectuar testes a determinados comportamentos.

O papel de guarda-redes (*Goalie*) tem como objectivo defender a baliza dos remates dos adversários. Este papel tem um sistema de auto-localização rudimentar, utilizando os postes da baliza e a linha de fundo para se localizar. Com esta auto-localização e integrando as sucessivas posições da bola é possível calcular a sua velocidade. O cálculo da velocidade da bola permite determinar se a bola se afasta, se vem em direcção à baliza ou se vem em direcção à baliza e não a intercepta.

Este papel tem como base uma máquina de estados simples como ilustra a figura 5.2. O estado *gsDefend* executa o comportamento *GoalieDefend*. Este comportamento calcula a intercepção da direcção da bola com a linha de fundo e caso essa intercepção esteja dentro da baliza move-se para esse ponto. O estado *gsKickAway* tem como objectivo afastar a bola da baliza, movendo-se para ela

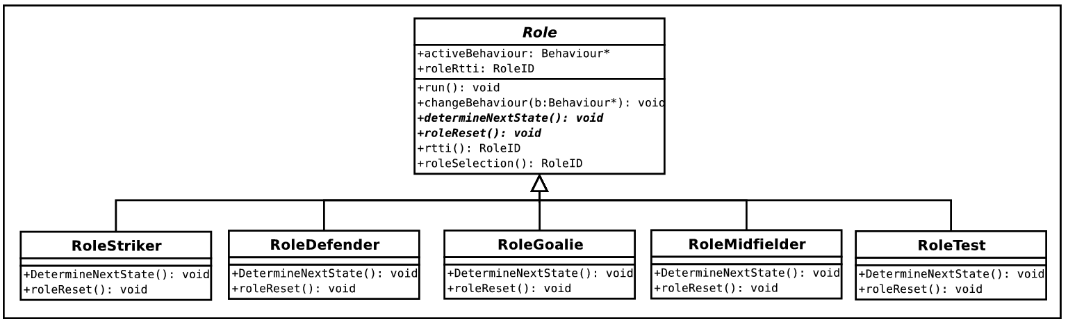

Role
+activeBehaviour: Behaviour*
+roleRtti: RoleID
+run(): void
+changeBehaviour(b:Behaviour*): voi
+determineNextState(): void
+roleReset(): void
+rtti(): RoleID
+roleSelection(): RoleID
RoleStriker
+DetermineNextState(): void
+roleReset(): void
RoleDefender
+DetermineNextState(): void
+roleReset(): void
RoleGoalie
+DetermineNextState(): void
+roleReset(): void
RoleMidfielder
+DetermineNextState(): void
+roleReset(): void
RoleTest
+DetermineNextState(): void
+roleReset(): void

Figura 5.1: Diagrama classes com os papéis usados na versão 2

e chutando-a. Este estado é activado quando a bola está *n* ciclos perto do guarda-redes. Este estado também tem um limite de ciclos e caso esse limite seja alcançado, volta para o estado *gsDefend*. O estado *gsRecalibrate* é um estado que deveria servir para o reposicionamento do guarda-redes, quando este se perde. Actualmente o estado de re-calibração não está implementado.

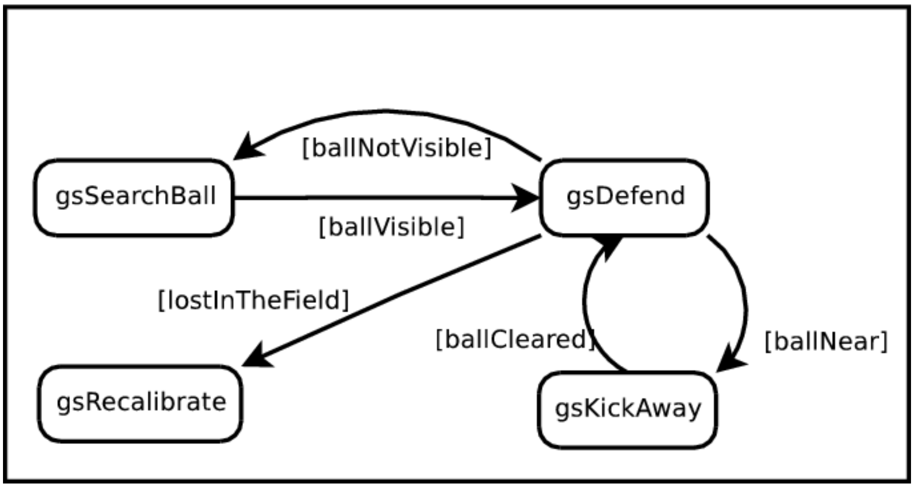

[ballNotVisible]
gsSearchBall
gsDefend
[ballVisible]
[lostInTheField]
[ballCleared]
[ballNear]
gsRecalibrate
gsKickAway

Figura 5.2: Diagramas de estados do papel *Goalie*

O defesa (*Defender*) [49] posiciona-se à frente da linha de grande área tem como principal objectivo bloquear a bola, caso venha no sentido da baliza. Este desloca-se paralelamente à linha de fundo de modo a impedir a passagem da bola. Funciona de modo semelhante ao *Goalie*, mas sem auto-localização.

O atacante (*Striker*) é um papel activo, que tem como objectivo interceptar a bola, driblar para a baliza adversária e chutar. Este papel foi implementado através de uma máquina de estados apresentada na figura 5.3.

A máquina de estados tem como objectivo apanhar a bola, conduzir e rematar a bola para a baliza. No caso de não ver a bola, o robô roda à procura da bola durante um certo período de tempo, depois procura a própria baliza, indo em direcção a ela até ficar à distância *d*, definida pelo utilizador. No caso de detectar a bola, o robô dirige-se a ela, até ter a bola ficar encaixada no sistema de retenção de bola. Depois roda em torno da bola, até se vericar o alinhamento o robô, a bola e a baliza adversária. Neste ponto estão reunidas as condições para começar o drible até à baliza adversária, evitando os

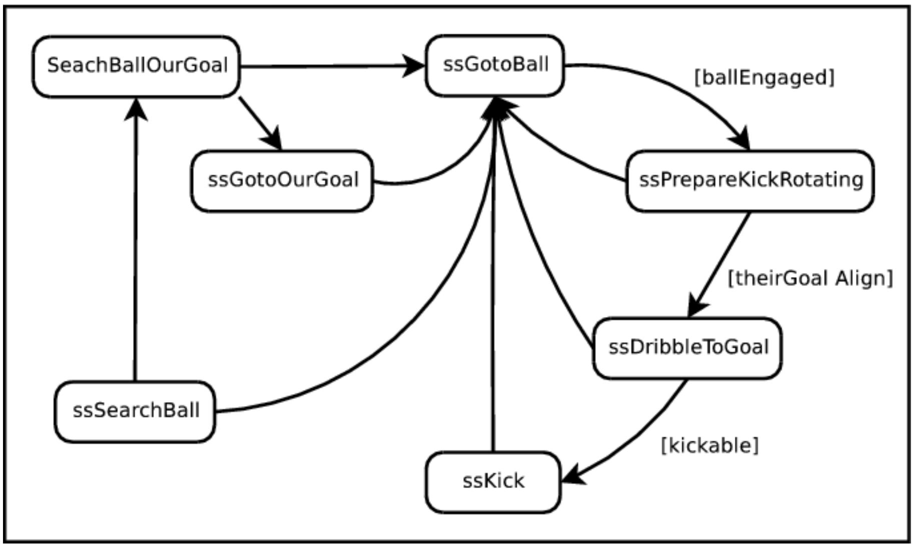


Figura 5.3: Diagramas de estados do papel *Striker*

adversários. Quando o robô estiver numa situação de remate efectua o comportamento para chutar a bola, que escolhe a melhor direcção para o remate.

Este papel também é responsável pelas reposições de bola a favor. Esta reposição consiste em manter o alinhamento robô, bola e baliza adversária. o robô mantém uma distância de 30 cm da bola. Com esta reposição de bola o robô fica pronto para avançar para a baliza adversária.

O *Midfielder* é papel usado durante as reposições de bola do adversário. Para evitar um remate à baliza ou uma saída de bola em direcção à baliza surgiu a necessidade de formar uma barreira. Como estes agentes não dispõem de auto-localização foi necessário desenvolver nova estratégia. A figura 5.4 ilustra a solução encontrada. Cada robô dirige-se em direcção à própria baliza. Quando a distância à própria baliza for inferior a três metros passa a dirigir-se para a bola ficando a dois metros da bola, distância miníma permitida pelas regras da MSL. Com os três robôs de campo a executarem este movimento e devido ao desvio de obstáculos eles formam uma barreira perfeita entre a bola e a própria baliza.

Após a descrição e implementação dos papéis, para finalizar este esquema de coordenação baseado em papéis é necessário definir que papel cada agente deve ter em cada instante. O algoritmo para a determinação de cada papel é baseado na distância à bola. Cada agente calcula a distância à bola de todos os agente. O agente mais próximo da bola é o atacante os restantes agentes de campo passam a ser defesas.

5.2 Versão 3/4

Para além das diferenças físicas descritas no capítulo 3 esta versão do agente CAMBADA dispõe de um módulo de auto-localização, baseado no algoritmo de localização [44] da equipa Tribot. Este algoritmo utiliza a detecção das linhas brancas [37] e minimiza o erro entre a a detecção feita e as linha de um campo virtual.

A versão 3/4 utiliza também um mecanismo de coordenação baseado em papéis. Os papéis (ver figura

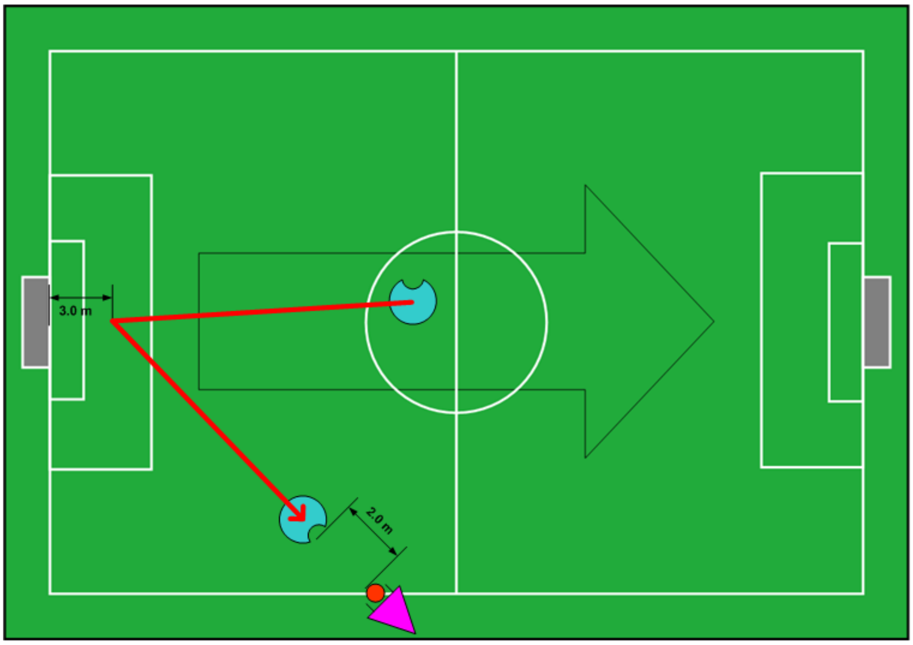


Figura 5.4: Formação da barreira

4.8) existentes nesta versão são o atacante (*Striker*), o guarda-redes (*Goalie*), o defesa (*Midfielder*). Para os reposicionamentos de bola são usados os papéis *Toucher*, *Replacer*, *Barrier*, *Penalty* e *Park*.

O *Striker*, nesta versão, também é implementado usando uma máquina de estados. Foi introduzida uma modificação em relação à versão anterior, a noção de consciência. Esta consciência é composta por um conjunto de condições que afectam a máquina de estados, independentemente do estado actual. A filosofia desta máquina de estados é semelhante à máquina de estados usado da versão 2. O facto da nova versão dispor de um módulo de auto-localização permite a simplificação da máquina de estados do atacante. Na figura 5.5 visualiza-se o diagrama de estados, com a utilização de uma consciência.

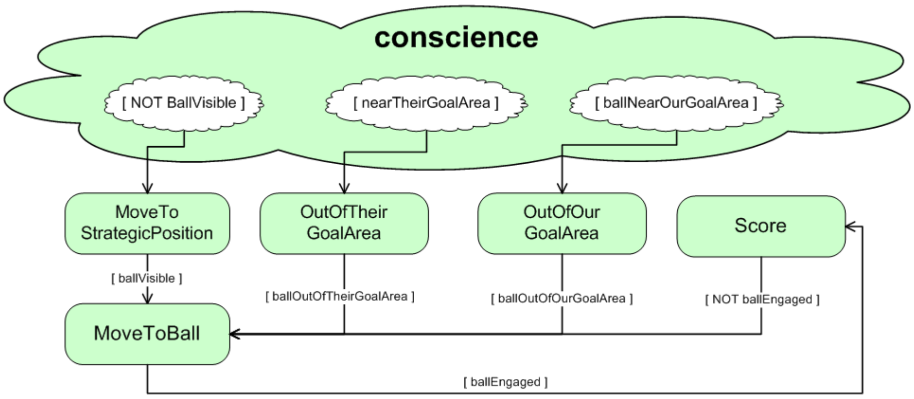

conscience
[NOT BallVisible]
[nearTheirGoalArea]
[ballNearOurGoalArea]
MoveTo StrategicPosition
OutOfTheir GoalArea
OutOfOur GoalArea
Score
[ballVisible]
[ballOutOfTheirGoalArea]
[ballOutOfOurGoalArea]
[NOT ballEngaged]
MoveToBall
[ballEngaged]

Figura 5.5: Diagrama de estados do *Striker*

A consciência é composta por três condições especiais. A primeira condição é despoletada pelo desconhecimento da posição da bola, originando uma transição para o estado *MoveToStrategicPosition*. A segunda condição é a proximidade à pequena-área do adversário. Caso o robô esteja ”perto” da área há uma transição para o estado *OutOfTheirGoalArea*. A terceira e última condição é a proximidade à própria pequena área, esta origina uma transição para o estado *OutOfGoalArea*.

MoveToStrategicPosition é o estado que permite ao atacante, quando não vê a bola, deslocar-se para a posição estratégica. No caso do número de agentes de campo ser superior a um, este agente desloca-se para uma posição estratégica mais avançada.

OutOfTheirGoalArea é o estado que evita que o atacante entre na pequena área , o que constitui uma falta segundo as regras da MSL. Por vezes a bola fica dentro da pequena área, neste o agente coloca-se à frente da bola de modo a evitar um remate por parte do guarda-redes (único robô com permissão a estar dentro da pequena área). No caso do atacante conseguir apanhar a bola que está dentro da área, roda sobre si próprio de modo a rematar em direcção à baliza.

OutOfOurGoalArea, como já foi referido anteriormente, é o estado que proíbe qualquer robô de entrar na pequena área, com a excepção do guarda-rede. Assim sendo, neste estado quando a bola se encontra na própria área, o agente afasta-se para fora da pequena área, evitando fazer falta e deixando espaço para o guarda-redes afastar a bola.

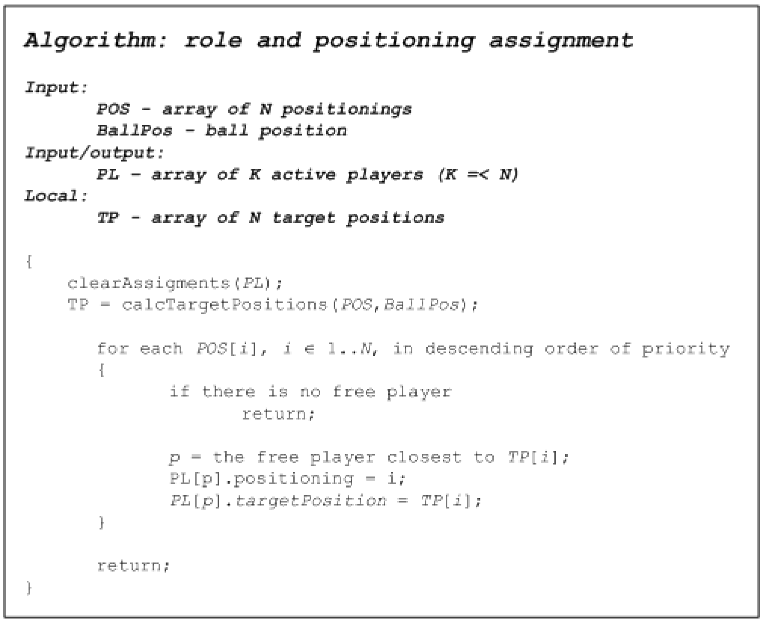
MoveToBall, como o próprio nome indica, é o estado que tem como objectivo apanhar a bola (*ballEngaged*). Usando uma *query* ao estado do mundo que devolve o risco de apanhar a bola, calculado através da posição da bola no campo. Este risco está catalogado em *SideBand*, *OurBand*, *TheirBand* e *NoDanger*. Quando a bola está localizada numa das zonas de perigo a movimentação para apanhar a bola é através de um ponto de aproximação, permitindo não atirar a bola para fora do campo. Durante a movimentação para o ponto de aproximação o agente orienta-se para a bola no caso de *TheirBand* e *OurBand*. Já no *SideBand* a orientação escolhida é a da bola. Quando não existe perigo, o agente pode apanha a bola de duas formas, dirigindo-se directamente para ela ou efectuando uma intercepção activa/passiva [45].

Score é o estado responsável por conduzir a bola para a baliza adversária, evitando os robôs adversários e rematar à baliza. Utiliza um método do estado do mundo chamado *kickable*, permitindo inferir que acção se deve escolher. O resultado *TryToScore* indica que a agente está em boas condições de efectuar um remate. *TooFar* indica que o agente está longe da baliza é necessita progredir no campo. Essa progressão é feita através do *Dribble* e a utilização de um campo de utilidades [50] específico do *Dribble*. *DeadAngle* indica que o ângulo de remate é muito apertado, logo para maximizar a probabilidade de marcar golo é necessária uma movimentação para o interior do campo. *Obstacle* resulta do factor de existir um robô à frente do agente impedindo o remate. Assim sendo, é necessário efectuar uma movimentação que evite este bloqueio.

A auto-localização permite tomar decisões diferentes dependendo da zona do campo, bem como melhorar a coordenação da equipa. Após a resolução do problema da auto-localização surge um novo desafio, o aumento das dimensões do campo. As dimensões aumentaram de $12m \times 8m$ para $18m \times 12m$, o que representa um aumento cerca de 50%. Este facto, aliado à falta de velocidade dos robôs da equipa CAMBADA, criou a necessidade de melhorar os mecanismos de coordenação.

Posicionamento Estratégico

O campo de grandes dimensões e a falta de velocidade dos robôs CAMBADA criou a necessidade de melhorar a distribuição da equipa de modo a maximizar a ocupação do campo. A solução adoptada foi baseada no algoritmo da equipa FCPortugal [51] da liga de simulação 2D, o *Situation Based Strategic Positioning* (SBSP) com *Dynamic Positioning and Role Exchange* (DPRE) [52,53].



```
Algorithm: role and positioning assignment

Input:
      POS - array of N positionings
      BallPos - ball position
Input/output:
      PL - array of K active players (K =< N)
Local:
      TP - array of N target positions

{
    clearAssigments(PL);
    TP = calcTargetPositions(POS,BallPos);

      for each POS[i], i ∈ 1..N, in descending order of priority
      {
            if there is no free player
                  return;

            p = the free player closest to TP[i];
            PL[p].positioning = i;
            PL[p].targetPosition = TP[i];
      }

      return;
}
```

Figura 5.6: Algoritmo de posicionamento da equipa CAMBADA

O algoritmo do posicionamento estratégico (formação) é implementado no módulo *Strategy* descrito no capítulo 4. Na figura 5.6 ilustra o algoritmo de posicionamento com troca dinâmica de posições. Este algoritmo permite calcular o posicionamento da equipa com base na posição da bola. O posicionamento é calculado através da soma da posicionamento base do agente com a posição da bola afectada de um factor de atracção. Este factor de atracção é composto pela componente dos *XX* e pela componente dos *YY*. A figura 5.7 ilustra quatro exemplos de posicionamentos, onde se visualiza os diferentes valores de atracção de cada agente. O posicionamento do canto superior esquerdo demonstra as posições de base da equipa CAMBADA.

Durante um jogo da MSL, por vezes os robôs devido a problemas eléctricos e/ou mecânicos necessitam de ser retirados de campo. No caso da equipa CAMBADA, a retirada de um robô causa a diminuição do número de robôs em campo. Então surgiu a necessidade de adaptar o algoritmo descrito em [52, 53], no sentido de ordenar por prioridades as posições estratégicas. A posição 1 (Figura 5.7: agente 6) localiza-se sempre a 50 cm da bola, naturalmente é ocupada pelo *Striker*, pois este é o papel atribuído ao agente que está em melhor posição para interceptar a bola. Esta precedência permite uma coordenação perfeita entre o posicionamento estratégico e a atribuição de papéis. A posição 2 (Figura 5.7: agente 2) é um posicionamento de defesa ou Líbero[1]. Com as posições 3, 4 e 5 (Figura 5.7: agentes 3, 4 e 5) pretende-se maximizar a ocupação do campo de modo a proteger a baliza. Estas três posições são ajustadas de jogo para jogo para adaptar a formação as características da equipa adversária. Estes ajustes permitem melhorar o desempenho global da formação pois permite a adaptação ao adversário.

---

[1] Designação para defesa com a função de recuperar a bola e ser o último defesa antes do guarda-redes

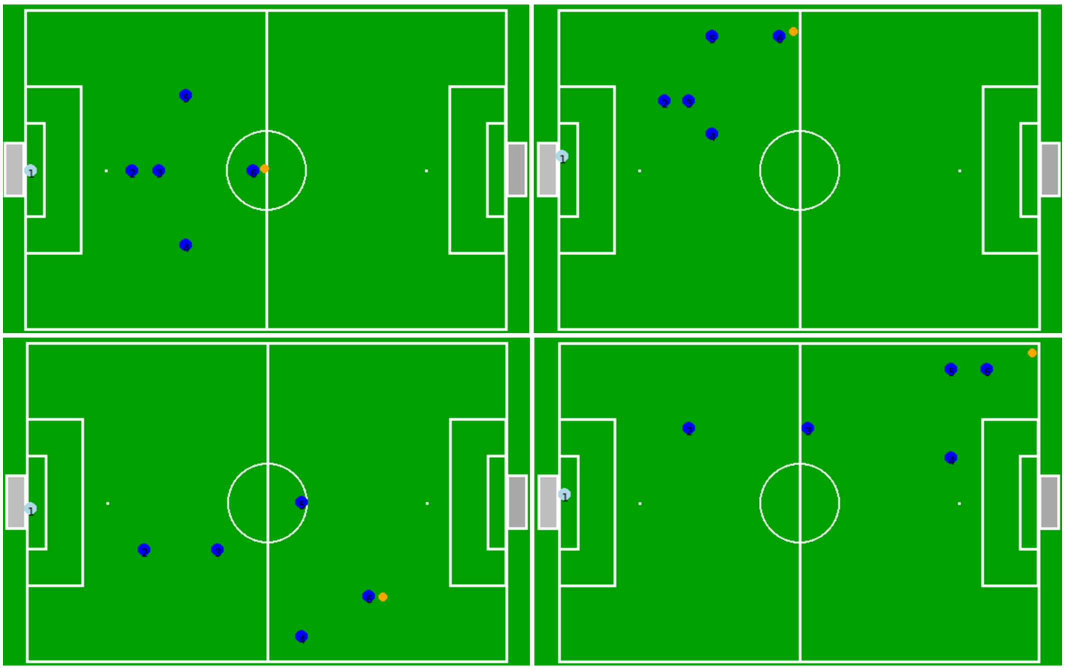


Figura 5.7: Quatro exemplos de posicionamento estratégico

Coach

O algoritmo da formação, como já foi referido anteriormente, permite a atribuição dinâmica de cada posição a cada agente. A posição é atribuída ao agente mais próximo dela. A decisão da atribuição de uma posição é decidida por um agente externo, o treinador (*Coach*). Este treinador recebe a informação partilhada por cada agente (localização, percepção da bola). Com esta informação consegue determinar as posições para cada agente e transmite-a. No caso da informação das posições não chegar aos agentes, pela comunicação não funcionar ou pelo *Coach* não estar activo, cada agente consegue determinar a sua própria posição.

O agente *Coach* pode determinar a formação adequada para um determinado momento do jogo. Para essa tomada de decisão usa a informação do resultado do jogo, do tempo de jogo e do número de agentes activos. Outro tipo de métricas usadas são o número de um determinado tipo de reposição de bola. No caso de a equipa ter muitos pontapés de baliza quer dizer que a sua baliza está a ser alvo de remates por parte do adversário. A discretização da posição da bola em zonas do campo e a distância média à bola por parte dos agentes são mais duas métricas que podem ser usadas para avaliar o uso da formação a usar.

5.3 Sumário

Neste capítulo foram explicados os mecanismo de coordenação da equipa CAMBADA. Esta explicação foi dividida em duas partes, a versão de 2 e versão 3/4. As principais diferenças destas duas versões assentam na ausência/presença de um módulo da auto-localização e no uso de visão Fontal / visão omni-direccional. Os mecanismos de coordenação baseiam-se em papéis e numa troca dinâmica entre eles, de uma forma implícita. Apresentou-se as modificações ao algoritmo SPSP/DPRE para satisfazer os requisitos das MSL.

Capítulo 6

Conclusão

6.1 Discussão de resultados

6.1.1 Síntese do trabalho desenvolvido

Com o objectivo de desenvolver e especificar uma arquitectura de controlo e de coordenação para a equipa de futebol robótico CAMBADA, o trabalho desenvolvido foi separado em duas fases.

Na primeira fase analisou-se as principais referências sobre agentes e suas arquitecturas e alguns trabalhos desenvolvidos pelas equipas da MSL (através de artigos e vídeos). Foi elaborada uma caracterização do ambiente de actuação, de tipos de agentes e de arquitecturas de agentes. De seguida apresentou-se um estudo dos principais sensores e actuadores utilizados pelos robôs das equipas MSL.

Na segunda fase efectuou-se um levantamento dos requisitos necessários para um aumento do desempenho da equipa CAMBADA, quer a nível do *hardware*, quer a nível de *software*. Após este levantamento foi necessário proceder a algumas alterações do robô CAMBADA (ver capítulo 3) e especificar a arquitectura de controlo, coordenação e *software*. A arquitectura especificada e desenvolvida baseia-se em papéis, comportamentos básicos e em posicionamento estratégico (formação). Este posicionamento estratégico foi uma adaptação do algoritmo SPSP-DPRE da equipa FCPortugal, da liga de simulação. Foi desenvolvido o agente treinador, responsável por determinar a posição estratégica, seleccionar a formação e recolher dados estáticos.

6.1.2 Artigos publicados

O trabalho desenvolvido no âmbito desta dissertação permitiu a publicação de quatro artigos em Conferências com Júri internacional, um artigo numa Revista Nacional e três artigos para os *Proceedings* do RoboCup (ver anexos).

Conferências com Júri internacional

RoboCup Middle Size Referee Box foi publicado na conferência *IADIS – Applied Computing, Salamanca/Espanha*. Este artigo reflectiu o trabalho da elaboração de uma nova arquitectura para a

RefereeBox [54] da MSL. A RefereeBox é o *software* que funciona como ponte entre as ordens dadas pelo árbitro humano e o mundo dos robôs.

Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots foi publicado na *13th Portuguese Conference on Artificial Intelligence, Guimarães/Portugal*. Este artigo retrata a auto configuração das comunicações *WiFi*. Esta auto configuração permite ajustar o instante da transmissão da informação por parte de cada agente, com o objectivo de eliminar colisões na transmissão.

An Omnidirectional Vision System for Soccer Robots foi publicado na *In Progress in Artificial Intelligence, edited by Springer Berlin / Heidelberg, Lecture Notes in Computer Science. Berlin, 2007*. Este artigo retrata a arquitectura do sistema de visão exposta no capítulo 4, secção 4.1. Esta arquitectura permite a detecção de objectos através de sensores radiais.

CAMBADA: Information Sharing and Team Coordination foi publicado na *Autonomous Robot System and Competitions: Proceedings of the 8th Conference, Aveiro/Portugal*. Este artigo descreve a arquitectura de partilha de informação e as metodologias de coordenação da equipa CAMBADA. Também é descrito a arquitectura de *software* e de decisão do agente CAMBADA.

Technical Description Papers – TDP

Os *Technical Description Papers* são artigos com descrição técnica e têm como objectivo demonstrar as inovações e os desenvolvimentos das equipa na fase de qualificação. Estes artigos são publicados nos *RoboCup Proceedings*.

CAMBADA'2006: Team Description Paper foi elaborado para a qualificação da equipa CAMBADA na MSL do RoboCup2006 em Bremen, Alemanha.

CAMBADA'2007: Team Description Paper foi elaborado para a qualificação da equipa CAMBADA na MSL do RoboCup2007 em Atlanta, Estados Unidos.

CAMBADA'2008: Team Description Paper foi elaborado para a qualificação da equipa CAMBADA na MSL do RoboCup2008 em Suzhou, China.

Revistas nacionais

The Base Station Application of the CAMBADA Robotic Soccer Team foi publicado na *Revista DETUA, Volume 5, Nº1, Novembro de 2008*. Este artigo retrata o trabalho desenvolvido na *BaseStation* na equipa de Futebol Robótico. A *BaseStation* é o *software* que permite a tradução e a comunicação das ordens recebidas pela Referee-Box, para os agentes CAMBADA. Permite visualizar a percepção que os agentes têm do ambiente.

Artigos Submetidos

Coordinated Action in Middle-Size Robotic Soccer foi submetido para *2009 IEEE International Conference on Robotics and Automation (ICRA)* e aguarda o resultado da avaliação. Este artigo retrata o trabalho desenvolvido na coordenação, partilha/integração de informação. São apresentados os resultados da competição RoboCup2008 e o desempenho da arquitectura e dos algoritmos descritos. Este desempenho é avaliado através de ficheiros de *log* e da análise de vídeos de jogos.

6.1.3 Análise de resultados

Nesta secção analisa-se os resultados das participações das competições nacionais e internacionais da equipa de futebol robótico CAMBADA.

O trabalho desenvolvido na definição e implementação da arquitectura de *software* e na arquitectura do agente CAMBADA permitiu um aumento do desempenho da equipa CAMBADA, que culminou na vitória no RoboCup208 em Suzhou, China.

As figuras 6.1 e 6.2 ilustram a evolução do desempenho da equipa CAMBADA nos torneios de futebol robótico. Este desempenho é avaliado através do número médios de golos marcados e sofridos por jogo.

Na figura 6.1 visualiza-se o desempenho da equipa CAMBADA nas competições nacionais. Este desempenho é avaliado através do número médio de golos marcados e sofridos.

O aumento do número médio de golos do Robótica 2005 para o Robótica 2006 deve-se à utilização do papel de atacante, desenvolvido no âmbito da implementação da arquitectura de coordenação e controlo da equipa CAMBADA. No Robótica 2007 foi utilizado pela primeira vez a auto-localização. Assim sendo os algoritmos de remate à baliza passaram a utilizar a auto-localização para determinar a posição da baliza. Este facto resultou num decréscimo do número médio de golos. Em 2008, já com os algoritmos de auto-localização estabilizados, foi possível melhorar o desempenho do atacante nas várias situações de jogo (apanhar a bola junto às linha, jogadas estudadas, etc).

Na figura 6.1 pode-se visualizar um decréscimo gradual do número médio de golos sofridos por jogo. Note-se que nos torneios Robótica 2007 e 2008 o número médio de golos sofridos por jogo é apenas de 0.5 e 0.3 golos. Este facto deve-se à utilização do posicionamento estratégico/formação permitindo travar a progressão do adversário bem como a rápida recuperação de bola.

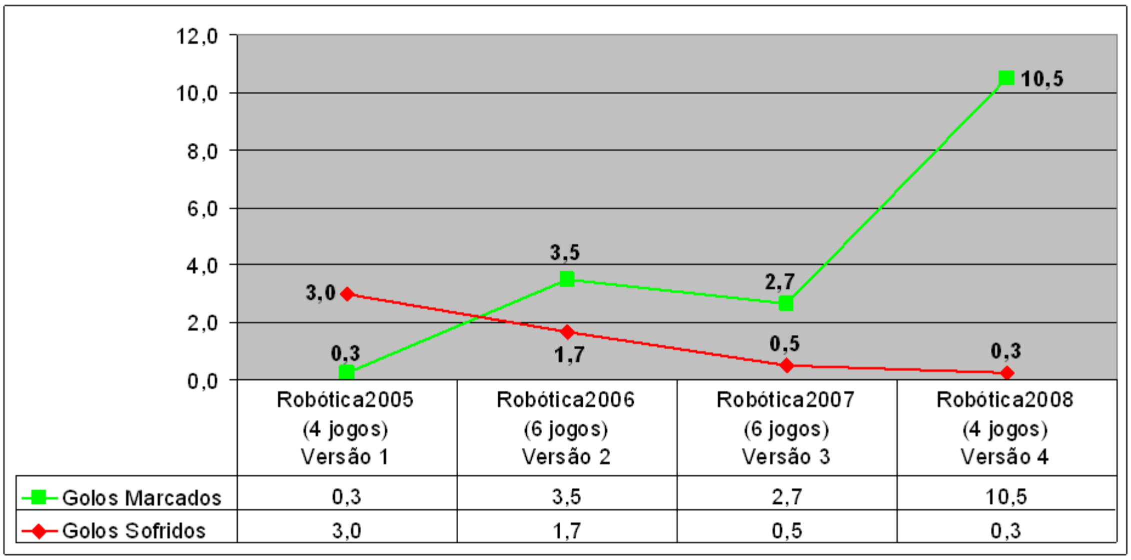


Figura 6.1: Número médio de golos marcados e sofridos por jogo em competições Nacionais

A 6.2 permite analisar o desempenho da equipa CAMBADA em torneios internacionais. Nesta figura pode-se visualizar o aumento de desempenho ao longo das competições internacionais. A partir do RoboCup 2007 houve um aumento de desempenho da equipa, quer a nível de golos marcados, quer a nível de golos sofridos. Este facto deve-se ao uso de auto-localização que permitiu a implementação do posicionamento estratégico/formação, bem como uma melhoria na coordenação da equipa (atribuição de papéis e jogadas estudadas).

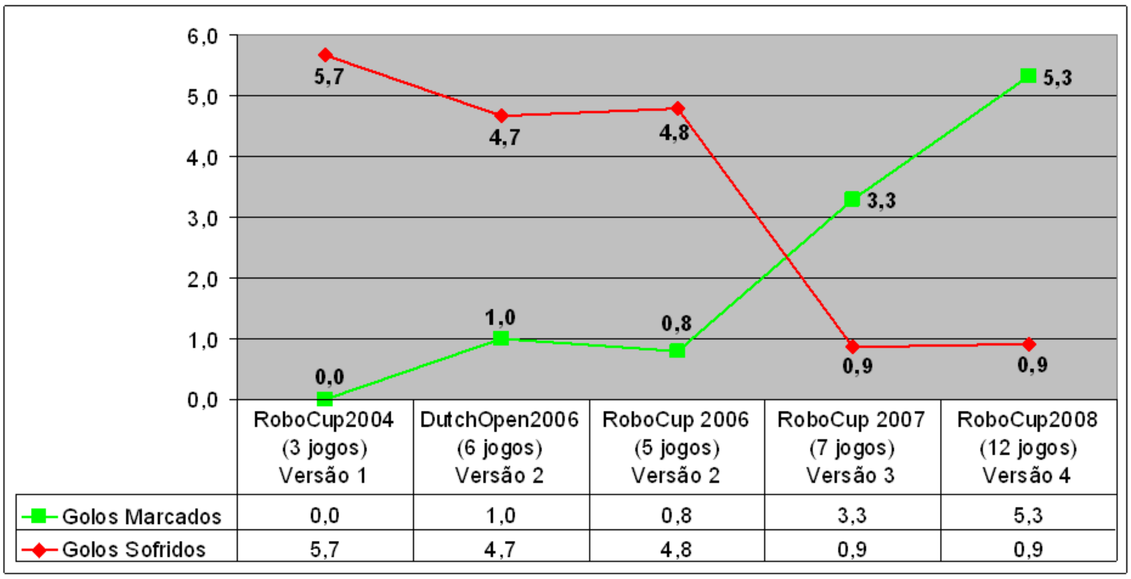


Figura 6.2: Número médio de golos marcados e sofridos por jogo em competições Internacionais

RoboCup 2008

Durante o RoboCup 2008, em Suzhou (China), a equipa CAMBADA sagrou-se campeã mundial na categoria Middle Size League (MSL). Na tabela 6.1 visualizar-se o desempenho da equipa CAMBADA. Num total de 13 jogos a equipa ganhou 11 e perdeu apenas 2, marcando um total de 73 golos e sofrendo 11. A equipa CAMBADA obteve o melhor *goal average* do torneio.

Fase	**Jogos**	**Vitórias**	**Empates**	**Derrotas**	**Golos Marcados**	**Golos Sofridos**
Grupos 1	5	3	0	0	41	2
Grupos 2	4	3	0	1	16	3
Grupos 3	2	1	0	1	5	2
Semi-final	1	1	0	0	4	3
Final	1	1	0	0	7	1
Total	13	11	0	2	73	11

Tabela 6.1: Resultados RoboCup2008

Com o objectivo de analisar a posse de bola durante o RoboCup 2008, recolheu-se os ficheiros de *log* dos jogos realizados, contendo as várias posições da bola. Para se efectuar a análise dividiu-se o campo em 12 partes iguais e enquadrou-se cada posição da bola numa das 12 zonas do campo. A figura 6.3 ilustra a localização da bola segundo as zonas definidas. Através da figura 6.3 é possível observar-se que a bola localiza-se 27% no próprio meio campo e 73% no meio campo adversário.

Analisando os resultados pode-se concluir que a combinação dos papéis *Striker* e *Midfielder*, com o posicionamento estratégico/formação permitiu à equipa CAMBADA superiorizar-se em termos de

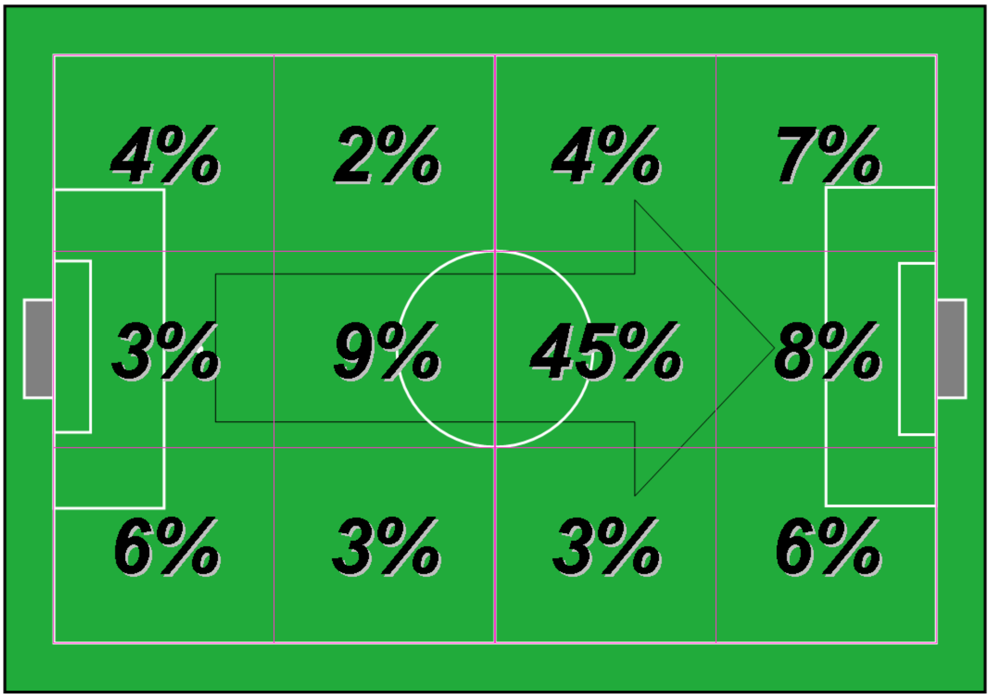


Figura 6.3: Localização da bola

tendência de jogo. Assim, estando a bola mais perto da baliza adversária aumenta a probabilidade de se marcar um golo. Esta conclusão é suportada pelo elevado número de golos marcados pela equipa CAMBADA.

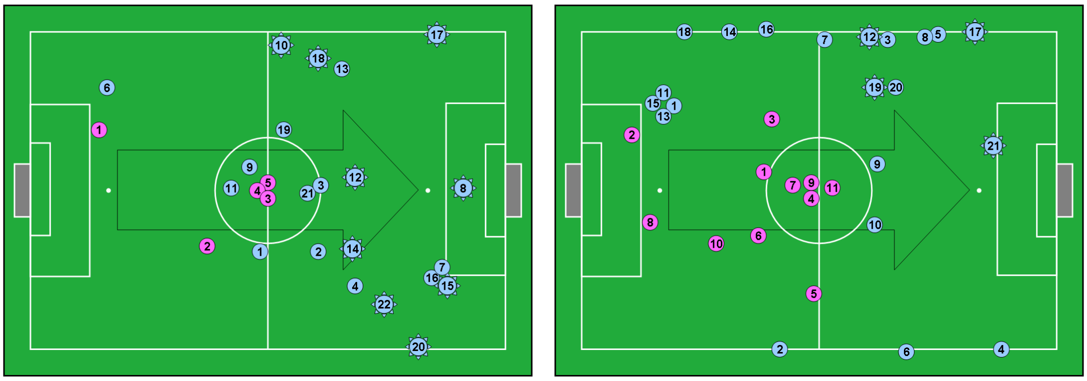
A figura 6.4 permite visualizar os remates sofridos e efectuados durante os jogos contra as equipas: MRL (*Azad University of Qazvin, Iran*), SCUT 100Steps (*South China University of Technology, China*), CoPS (*University of Stuttgart, Germany*) e TechUnited (*Eindhoven University of Technology, Netherlands*). Os remates efectuados são representados por círculos azuis, enquanto os remates sofridos são representados pelos círculos magenta. Os remates que originaram golos são representados através de um sol, com a respectiva côr.

Através da análise da figura 6.4 pode concluir-se que a equipa CAMBADA é a mais rematadora, sendo os remates efectuados predominantemente no campo do adversário.

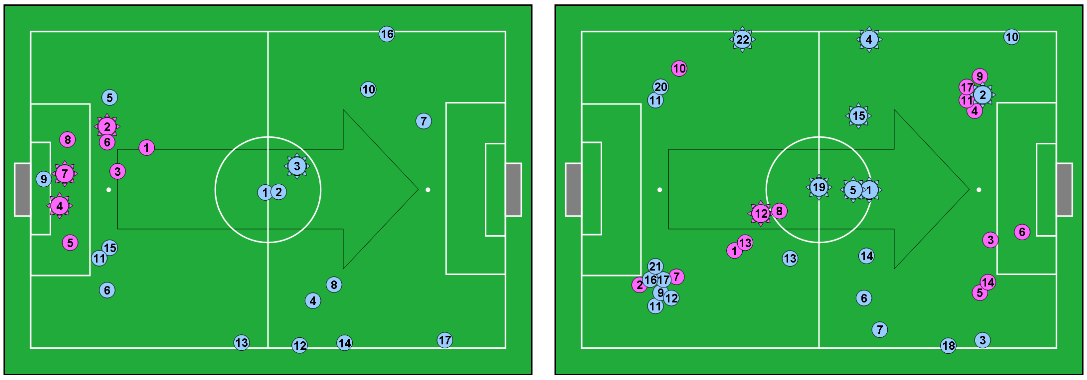
Com base nos ficheiros de *log* foi possível calcular a distância à bola do robô mais próximo. Durante o RoboCup 2008 a valor médio da distância miníma à bola foi de 1.246 ± 0.325 metro Como a distância miníma é menor a 1.3 metros, valor este inferior à velocidade máxima do robô CAMBADA, permite que a equipa controle a progressão da bola. Este resultado permite avaliar positivamente a implementação/utilização do posicionamento estratégico/formação da equipa CAMBADA.

6.1.4 Notas Finais

Na final do Robótica 2007, após o empate a zero no tempo regulamentar, foi necessário decidir o torneio através da marcação de penaltis. A arquitectura de controlo/coordenação não previa a situação de penaltis. Durante a escolha da baliza e da ordem de marcação dos penaltis, foi necessário implementar este papel, em menos de 5 minutos. Para cumprir os requisitos modificou-se o papel de atacante



(a) Round Robin 2: CAMBADA 9 – SCUT 100Steps 0 (b) Round Robin 3: CAMBADA 4 – MRL 0



(c) Meia-Final: CAMBADA 4 – COPS 3 (d) Final: CAMBADA 7 – TechUnited 0

Figura 6.4: Remates executados e sofridos pela equipa CAMBADA

de modo a efectuar a marcação dos penaltis. A primeira equipa a marcar os penaltis foi o MINHO, marcando apenas dois em cinco. A equipa CAMBADA marcou três em três ganhando pela primeira vez o campeonato nacional.

O desafio técnico (*Technical Challenge*) [2] obrigatório para RoboCup 2008, consistia jogar com uma bola arbitrária (FIFA nº5). O desafio era composto por três tentativas onde eram utilizadas três bolas diferentes, que foram colocadas em locais aleatórios. O *software* da visão existente apenas estava preparado para detectar a bola através da sua cor laranja. Com o objectivo de resolver este desafio foi necessário desenvolver um novo *software* de visão, a *VisionThread* [55]. Do ponto de vista do agente apenas foi necessário utilizar a *VisionThread* em vez da visão convencional, sem efectuar nenhuma alteração nos módulos de integração, controlo e decisão do agente.

Os dois episódios relatados demonstram a modularidade e flexibilidade da arquitectura de *software*, controlo e coordenação, que foi especificada e implementada no âmbito desta dissertação.

Como conclusão final, importa salientar que a utilização da arquitectura especificada e implementada no âmbito desta dissertação foi um sucesso, culminando na vitória do RoboCup 2008.

6.2 Trabalho futuro

Após a especificação e implementação da arquitectura de controlo e de coordenação dos agentes CAMBADA e com extraordinário desempenho da equipa CAMBADA, torna-se inevitável defender os títulos alcançados em 2008, o bi-campeonato no Robótica 2008 e 1º lugar RoboCup 2008. Torna-se igualmente importante propor novas estratégias e algoritmos nas áreas de visão por computador e inteligência artificial.

Neste fase, a equipa CAMBADA necessita de novas ferramentas de *debug* e de *logging*. As ferramentas de *debug* e *logging* são necessárias para uma avaliação e uma validação dos algoritmos desenvolvidos, sejam eles de processamento dos sensores, integração no estado do mundo, de decisão e de coordenação de alto nível.

Com o objectivo de melhorar a arquitectura do agente especificada e desenvolvida com o presente trabalho seria importante definir uma nova linguagem para implementação de papéis. Esta linguagem permitiria o uso dos comportamentos implementados em *C/C++*. O principal requisito seria desenvolver papéis ou parte deles e reutilizá-los noutros papéis. Um requisito obrigatório desta linguagem seria o suporte para coordenação entre vários agentes. A geração desta linguagem deveria ser suportada por uma ferramenta gráfica. Nesta área existe trabalho desenvolvido pela equipa COPS da Universidade Estugarda[1] [18, 19] e pela empresa Gostai[2], com a sua linguagem URBI [56, 57].

Um ponto de evolução obrigatório para equipa é o uso de passes durante o jogo. Para cumprir este objectivo, o estado do mundo deve ser melhorado, aumentando a precisão da auto-localização e melhorando a integração da informação recolhida pelos sensores, nomeadamente a informação da posição e velocidade da bola. A detecção dos adversários é importante pois permitirá detectar as zonas livres do campo. O robô CAMBADA também precisará de algumas alterações, nomeadamente, um aumento de velocidade, um melhoramento no sistema de retenção da bola e a possibilidade de efectuar um remate rateiro da bola.

O aumento do campo, a ausência de um campo real e ausência de adversários, são três factores que tornam necessário o desenvolvimento de um simulador. Este simulador, numa primeira fase, deve apenas simular a recolha de informação sensorial e de actuação, mas, numa segunda fase, deverá simular os sensores, nomeadamente a imagem recolhida pelas câmaras.

O agente treinador (*Coach*) não deverá apenas escolher a posição estratégia de cada agente. Por forma a optimizar o desempenho da equipa CAMBADA durante o jogo, o *Coach* deverá adaptar o modelo de jogo da equipa de acordo com o desenrolar do jogo.

---

[1] `http://robocup.informatik.uni-stuttgart.de/`

[2] `http://www.gostai.com/`

Bibliografia

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The Robot World Cup Initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM New York, NY, USA, 1997.

[2] MSL Technical Committee. Msl rules 2008.

[3] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[4] M. Arbatzat, S. Freitag, M. Fricke, R. Hafner, C. Heermann, K. Hegelich, A. Krause, J. Kruger, M. Lauer, M. Lewandowski, et al. Creating a Robot Soccer Team from Scratch: the Brainstormers Tribots. In *RoboCup-2003-Proceedings of the International Symposium*, 2003.

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

[6] T. Buchheim, G. Kindermann, R. Lafrenz, H. Rajaie, M. Schanz, F. Schreiber, O. Zweigle, and P. Levi. Team Description Paper 2004 CoPS Stuttgart. *RoboCup 2004: Robot Soccer World Cup VIII*.

[7] Sash. Robot from scratch. *y*, 1(2):12–67, mAU 2323. uu.

[8] B. Cunha, J. Azevedo, N. Lau, and L. Almeida. Obtaining the Inverse Distance Map from a Non-SVP Hyperbolic Catadioptric Robotic Vision System. *Lecture Notes in Computer Science*, 5001:417–424, 2008.

[9] W. Richert, B. Kleinjohann, M. Koch, A. Bruder, S. Rose, and P. Adelt. The Paderkicker Team RoboCup 2006.

[10] V. Cerqueira, A. Dias, N. Matos, JM Almeida, A. Martins, and EP Silva. ISEPorto Robotic Soccer Team: A New Player Generation.

[11] JL Azevedo, N. Lau, G. Corrente, A. Neves, MB Cunha, F. Santos, A. Pereira, L. Almeida, LS Lopes, P. Pedreiras, et al. CAMBADA'2008: Team Description Paper.

[12] JG Goorden and PP Jonker. TechUnited Team Description.

[13] A.A.F. Nassiraei, Y. Takemura, A. Sanada, Y. Kitazumi, Y. Ogawa, I. Godler, K. Ishii, H. Miyamoto, and A. Ghaderi. Concept of Mechatronics Modular Design for an Autonomous Mobile Soccer Robot. In *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*, pages 178–183, 2007.

[14] *Tribots: Soccer Strategy*. http://www.ni.uos.de/fileadmin/user_upload/tribots/Research/Kooperation.pdf.

[15] Stefan Welker Sascha Lange, Christian Müller. *Tribots: Architecture*. http://www.ni.uos.de/fileadmin/user_upload/tribots/Research/Architektur.pdf.

[16] Stefan Welker Sascha Lange, Christian Müller. *Behavior-Based Approach*. http://www.ni.uos.de/fileadmin/user_upload/tribots/Research/Behaviors.pdf.

[17] M.J. Wooldridge. *Reasoning About Rational Agents*. MIT Press, 2000.

[18] O. Zweigle, R. Lafrenz, T. Buchheim, H. Rajaie, F. Schreiber, and P. Levi. Cooperative agent behavior based on special interaction nets. In *Intelligent Autonomous Systems*, volume 9, 2006.

[19] R. Lafrenz, O. Zweigle, UP Kappeler, H. Rajaie, A. Tamke, T. Ruhr, M. Oubbati, M. Schanz, F. Schreiber, and P. Levi. Major Scientific Achievements 2006-CoPS Stuttgart registering for world championships in Bremen. *RoboCup 2006: Robot Soccer World Cup X Preproceedings*.

[20] M. Lotzsch, J. Bach, H.D. Burkhard, and M. Jungel. Designing agent behavior with the extensible agent behavior specification language XABSL. In *7th International Workshop on RoboCup*. Springer, 2003.

[21] Maxon motors home page. http://www.maxonmotor.com.

[22] *Wikipedia - CoilGun*. http://en.wikipedia.org/wiki/Coilgun.

[23] *Ubuntu home page*. http://www.coilgun.eclipse.co.uk/.

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

[25] Microchip home page. http://www.microchip.com.

[26] *Bosch CAN home page*. http://www.can.bosch.com.

[27] R.B. GmbH. CAN Specification. *Robert Bosch GmbH, Postfach*, 50.

[28] J. A. Fonseca. L. Almeida, P. Pedreiras. The ftt-can protocol: Why and how. *IEEE Transactions on Industrial Electronics*, 49(2), December 2002.

[29] *Linux home page*. http://www.linux.org.

[30] *Debian home page*. http://www.debian.org.

[31] *Intel Core2Duo home page*. http://www.intel.com/products/processor/core2duo/index.htm.

[32] *Ubuntu home page*. http://www.ubuntu.com/.

[33] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L.S. Lopes. Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 876–886, 2004.

[34] F. Santos, L. Almeida, and L.S. Lopes. Self-configuration of an adaptive TDMA wireless communication protocol for teams of mobile robots. In *Emerging Technologies and Factory Automation. ETFA 2008. IEEE International Conference on*, pages 1197–1204, 2008.

[35] L. Seabra Lopes Santos F., L. Almeida. Adaptive wireless communication for teams of mobile robots: a position paper. *Workshop on Real Time Networks, Prague, Czech Republic*, July 2008.

[36] F. Santos G. Corrente L. Almeida N. Lau L. Seabra Lopes. Self-configuration of an adaptive tdma wireless communication protocol for teams of mobile robots. *13th Portuguese Conference on Artificial Intelligence, Guimarães, Portugal*, December 2007.

[37] A. Merke, S. Welker, and M. Riedmiller. Line based robot localization under natural light conditions. In *European Conference on Arificial Intelligence Machine Learning (ECAI)*, 2004.

[38] A.J.R. Neves, G.A. Corrente, and A.J. Pinho. An omnidirectional vision system for soccer robots. In *Lecture Notes in Computer Science - Progress in Artificial Intelligence*. Springer Berlin / Heidelberg, 2007.

[39] A.J.R. Neves, D.A. Martins, and A.J. Pinho. A hybrid vision system for soccer robots using radial search lines. *Robótica 2008: 8th Conference on Autonomous Robot Systems and Competitions*, 2008.

[40] J. E. Bresenham. *A linear algorithm for incremental digital display of circular arcs.* CA CM.

[41] J. E. Bresenham. *Algorithm for computer control of a digital plotter.* IBM Systems J.

[42] R.C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[43] The Base Station Application of the CAMBADA Robotic Soccer Team. Nuno figueiredo, antónio neves, nuno lau, josé azevedo, artur pereira and gustavo corrente. In *REVISTA DO DETIUA*, NOVEMBRO 2008.

[44] M. Lauer, S. Lange, and M. Riedmiller. Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. *LECTURE NOTES IN COMPUTER SCIENCE*, 4020:142, 2006.

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

[46] R.E. Kalman. A new approadh to linear filtering and prediction problems. In *Journal of Basic Engineering*, 1960.

[47] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality af A*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[48] *Bash home page*. http://www.gnu.org/software/bash/.

[49] José Rocha. Implementação de defesa para a equipa de futebol robótica cambada. Master's thesis, Universidade de Aveiro, 2006.

[50] Nelson Filipe. Individual and coordinated decision functionalities for the cambada team. Master's thesis, Universidade de Aveiro, 2008.

[51] Luís Paulo Reis and Nuno Lau. Fc portugal team description: Robocup 2000 simulation league champion. In *RoboCup-2000: Robot Soccer World Cup IV, Springer Verlag Lecture Notes in Artificial Intelligence*.

[52] Nuno Lau Luís Paulo Reis and Eugénio C. Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In Jan Wendler Markus Hannebauer and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent System: From RoboCup to Real-World Applications*, Berlin, 2001. Springer Lecture Notes in Artificial Intelligence.

[53] 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*, Berlin, 2001. Springer Verlag Lecture Notes in Artificial Intelligence.

[54] Gustavo A. Corrente. Robocup middle size league referee box. *IADIS international Conference: Applied Computing 2007, Salamanca, Espanha*, 2007.

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

[56] J. Baillie. Urbi: A universal language for robotic control. *International Journal of Humanoid Robotics*, 2004.

[57] J.C. Baillie. URBI: towards a universal robotic low-level programming language. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 820–825, 2005.

Apêndice A

CAMBADA'2006: Team Description Paper

CAMBADA'2006: Team Description Paper

L. Almeida, J.L. Azevedo, G. Corrente, M.B. Cunha, A. Ferdowsi,
J.P. Figueiredo, P. Fonseca, S. Lopes, R. Marau, N. Lau, P. Pedreiras, A. Pereira,
A. Pinho, J. Rocha, F. Santos, L. Seabra Lopes, V. Silva, J. Vieira

Transverse Activity on Intelligent Robotics
IEETA/DET – Universidade de Aveiro
3810-193 Aveiro, Portugal

Abstract. The CAMBADA middle-size robotic soccer team is described in this paper for the purpose of qualification to RoboCup'2006. This team was designed and developed by the authors, from scratch, in the last three years. The players, completely built in-house, incorporate several innovations at the hardware level, particularly the sensing and computational subsystems. At the software level, cooperative sensing uses a real-time database implemented over a real-time Linux kernel. Previous experience of the team in the simulation league has been highly relevant. The paper focuses on recent advances on vision, localization and monitoring/debugging software as well as a new ultrasound-based localization system.

1 Introduction

CAMBADA[1] is the RoboCup middle-size league soccer team of the University of Aveiro. This project, started officially in October 2003, is funded by the Portuguese research foundation (FCT) [2]. CAMBADA participated in RoboCup'2004 and in the last two editions of the Portuguese Robotics Festival (RoboCup'2004 and '2005).

The previous CAMBADA Team Description Paper [2], prepared for RobCup'2004, provides a detailed overview of the team, as it was initially designed and developed. Some aspects of its design were demonstrated in RoboCup'2004 while others were implemented since then. The present paper provides a shorter overview of the project and then focuses on recent developments.

The CAMBADA players were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter, which allows for a team of 5 robots, according to the rules. The mechanical structure of the players is layered and modular (Figure 1). Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromechanical kicker, are attached to an aluminium plate placed 8 cm above the floor. The second layer contains the control electronics. The third layer contains a computer, at 22.5 cm from the floor. The players are capable of holonomic motion, based on three omni-directional roller wheels [5].

---

[1] CAMBADA is acronym of *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*; 'cambada' is also a Portuguese word for 'band' or 'mob'.

[2] Project POSI/ROBO/43908/2002, partially funded by FEDER.

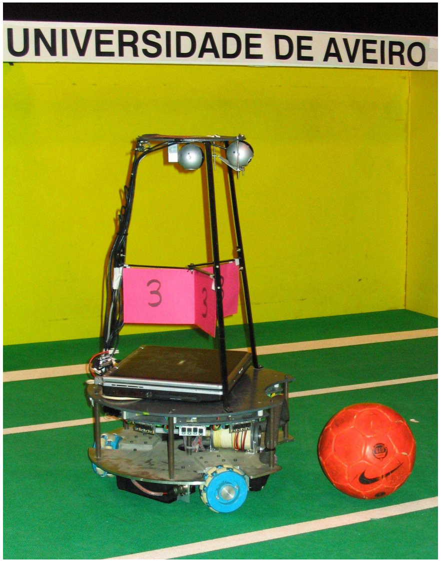


Fig. 1. One of the CAMBADA players

The main sensors in each player are two webcams, both equipped with wide-angular lenses and installed at approximately 80cm above the floor. Both cameras deliver 320x240 YUV images at a rate of 20 frames per second (fps). One of the cameras faces the field orthogonally, enabling to capture a 360 degrees view around the robot, approximately with a 1m radius. This so-called omni-directional vision system is used for obstacle avoidance and ball handling.

The other camera points forward in the direction of the front of the robot, with 57º inclination of with respect to its vertical axis. This frontal system is used to track the ball at medium and long distances, as well as the goals, corner posts and players. All the objects of interest are detected using simple color-based analysis, applied in a color space obt ained from the YUV space by computing phases and modules in the UV plane.

The robots computing system architecture follows the fine-grain distributed model [6] where most of the elementary functions, e.g. closed-loop control of complex actuators, are encapsulated in small microcontroller-based nodes, connected through a network. A PC-based node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. USB or Firewire (Fig. 3). For this purpose, Controller Area Network (CAN), a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [3]. The communication among robots uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment.

The software system in each player is distributed among the various computational units. High level functions run on the computer, in Linux operating system with RTAI (Real-Time Application Interface). Low level functions run partly on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1,2,4,7,8] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

The high-level processing loop starts by integrating perception information gathered locally by the player. This includes information coming from the vision processes, which is stored in a Local Area of the RTDB, and odometry information coming from the holonomic base via FTT-CAN. After integration, the world state can be updated in the shared area of the RTDB. The next step is to integrate local information with information shared by teammates. This will be the basis for taking decisions according to a finite state machine. Each state is characterized by the behavior pattern that is executed. A very basic coordination mechanism is currently supported. According to this mechanism, the player that takes control of the ball is the player closest to the ball. Other players take strategic positions in the field based on their distances to the goals. We expect to improve the coordination mechanism as soon as localization capabilities are fully evaluated. This also depends on the availability of monitoring and debugging tools, which are under development.

2 Real-time vision architecture

A modular multi-process architecture was adopted for the vision software subsystem (Figure 2) [7]. For each camera, one process is automatically triggered whenever a new image frame is ready for dowload. The frame data are placed in shared image buffers, which are afterwards analyzed by the object detection processes, generically designated by proc_obj:*x*, $x=\{1,2,\ldots \boldsymbol{N}\}$. These processes are encapsulated in separate Linux processes. Once started, each process gets a pointer to the most recent image frame available and starts tracking the respective object. Once finished, the resulting information (e.g. object detected or not, position, confidence) is placed in the real-time database. This database may be accessed by any other processes on the system, particularly for world state update.

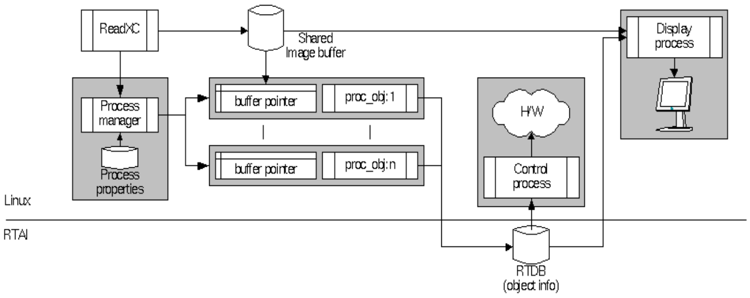


Fig. 2. Vision subsystem software architecture

The activation of the distinct image-processing activities is carried out by a process manager. Each object tracking process (i) is associated with a period (P_i) and a phase (φ_i), expressed as integer number of image frames. For every frame f, the process manager activates all the processes that verify $[(f-\varphi_i)\%\ P_i\ ]=0$. This allows allocating periods according to the specific attributes of each object (e.g., the ball is highly dynamic and is tracked in every frame while the relative goal position is less dynamic and can be tracked every four frames) as well as to de-phase them in the time domain, minimizing the mutual interference and consequently their response time and jitter.

Scheduling of vision related processes relies on the real-time features of the Linux kernel, namely the FIFO scheduler and priorities in the range 15-50. At this level, Linux executes each process to completion, unless the process blocks or is preempted by other process with higher real-time priority. This ensures that the processes are executed strictly according to their priority with full preemption. The real-time features of Linux are sufficient at this time-scale (periods multiple of 50ms).

3 Information Integration and Localization

Localization in the play field is a very basic requirement for implementing advanced coordination and cooperation strategies. Localization includes self-localization and localization of the ball and players. Localization is the main outcome of local and team-level information integration. As expected, odometry information is not enough to maintain sufficiently accurate localization information in CAMBADA [4]. After long distances or through collisions between players, it is very easy to reach positional error levels not acceptable for team coordination purposes. In collision-free runs of 100 m, we verified that the positional error grows roughly linearly with the distance travelled by the player. The error is around 1.5% to 2.% of the distance. Therefore, position errors of 2m can easily occur.

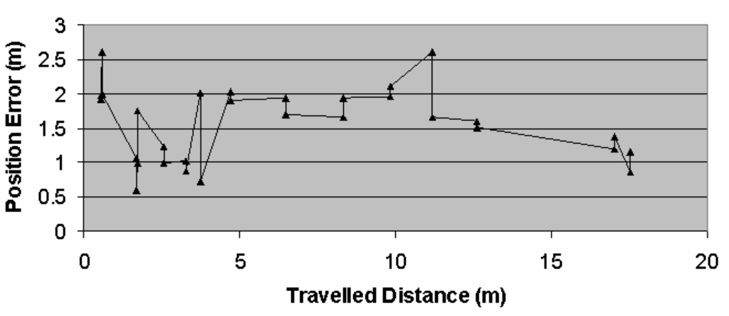


Fig. 3. Effect of opportunistic vision-based calibration (example run)

Localization in the currently working CAMBADA team is based on odometry information, updated in each iteration of the control loop, and calibrations performed based on vision information. The calibration mechanisms can be grouped as follows:

- Opportunistic, based on a single landmark (goal, corner post, line) – not enough to derive the player's position and orientation but enables calibration.
- Opportunistic, based on two successively seen landmarks – enables to calculate position/orientation; error inherent to the vision system only.
- Active – The player actively searches for two landmarks, e.g. by performing a full turn around itself.

In opportunistic calibration, the vision-based positions/orientations are averaged with the internally kept values. Active localization is called in extreme situations, and the obtained values replace the previous values. When a change in internal values takes place, the new values are sent down to the odometry micro-controller.

While monitoring/debugging tools are being developed, we have been resorting to time-consuming "manual" evaluation experiments. These experiments show that the position and orientation errors can be reduced to acceptable levels using the methods enumerated above. Figure 3 shows the localization performance in one experiment, in which the initial position error was set to 2.24m. We see that, after running for around 17 meters and having performed 16 opportunistic calibrations, the position error was gradually reduced to ~1m.

4 Ultrasound-based Localization

In parallel with the vision-based localization capabilities described above, we have been developing an alternative/complementary localization system based on ultrasound sensors [9]. Advantage is taken from the fact that the goalkeeper is near the goal, being easy to obtain an absolute position in the field using visual information. The goalkeeper has one ultrasound emitter that transmits a pulse in a periodic way. The other robots have several ultrasound receivers that cover all the 360º around, and they reply a certain time after receiving the pulse from the goalkeeper. Each robot uses a different reply time in order to implement a time multiplexing of the answers. The goalkeeper knows the reply times of each robot, and in this way it can measure the propagation time of the sound to each robot and from this calculate the distance of each robot. The goalkeeper also has two ultrasonic sensors in order to measure the angle of the signal received from each robot. With these two values, it computes the *x*, *y* coordinates of the robots in the field.

The ultrasound signals are processed using the DSP from Texas Instruments 2812. An initial proof-of-concept prototype was developed using simple algorithms. We use chirps as the transmitted signal and matched filters to detect the pulses. This way we managed to solve some of the problems related to multipath propagation and it is also possible to share the acoustic channel using different chirp signals for each robot. The first field experiments showed that the system works in real conditions measuring the coordinates of the robots with good accuracy. The system has full room to improve the accuracy of the measures by only changing the signal processing algorithms.

5 Monitoring framework for multi-process/multi-agent systems

Cambada robots run several processes and at the same time they interact with each other. They operate autonomously, taking many decisions per second based on sensory information that changes dynamically and on shared information that is also subject to frequent changes. It is very hard to follow the robot’s reasoning based only on the external observation of its behavior. To aid this tuning and debugging process, a framework was developed to allow the visualization of the robots reasoning and synchronize it with the observed robot’s behavior.

Several constraints must be considered. The robot is executing several processes and in certain situations we should tune the behavior of the team as a whole instead of tuning individual robots. The framework is prepared to provide high-level information to the developer, useful for online observation, individual and team behavior tuning.

During execution, robots may send information to a socket or to a local logfile. The following debug data is attached to every item of information:

- Timestamp: Used to timeline the sequence and to synchronize information from several sources;
- Category: A tree of categories may be defined to better organize and visualize information. A certain tree or subtree of categories can be hidden/displayed;
- Level of detail: Useful to truncate the visualization at a certain level.

Several types of records are allowed, like text, bookmarks, video images, etc. To synchronize logfiles from different robots two options are available: Use of the regular clock of the PC with the inclusion of a NTP server in the team's base station PC and NTP clients in the robots or the use of the RTAI distributed clock available from the RTAI layer in Linux.

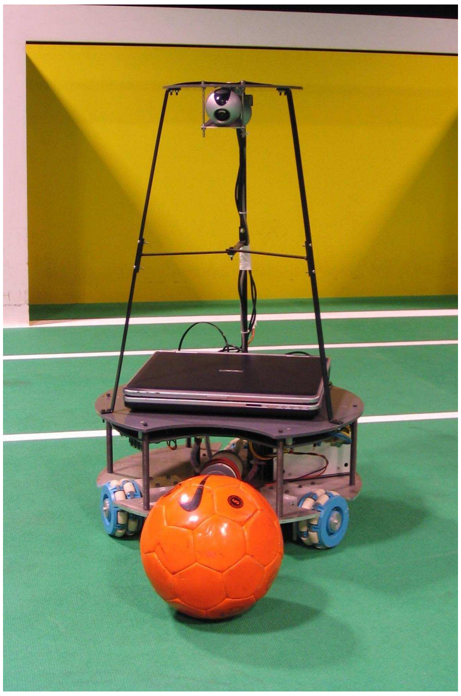
For file processing and reading, several features were implemented:

- Multiple file opening and managing;
- Time based interlace of records from the logfiles; this gives the user the feeling of one big logfile and allows to navigate the data on a unique time line.

An application is being developed for reading and analyzing logfiles.

References

1. Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes (2004) Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences -- ISCIS 2004: 19th International Symposium, Proceedings*, Aykanat, Cevdet; Dayar, Tugrul; Korpeoglu, Ibrahim (Eds.), Lecture Notes in Computer Science, Vol. 3280, p. 876-886.
2. Almeida, L., J.L. Azevedo, P. Bartolomeu, E. Brito, M.B. Cunha, J.P. Figueiredo, P. Fonseca, C. Lima, R. Marau, N. Lau, P. Pedreiras, A. Pereira, A. Pinho, F. Santos, L. Seabra Lopes, J. Vieira (2004) *CAMBADA: Team Description Paper, RoboCup Symposium: Papers and Team Description Papers* [CD].
3. Almeida, L., P. Pedreiras and J.A. Fonseca (2002) "FTT-CAN: Why and How", *IEEE Tran. Industrial Electronics*.
4. Bartolomeu, P., L. Seabra Lopes, N. Lau, A. Pinho, L. Almeida (2005) Integração de Informação na Equipa de Futebol Robótico CAMBADA, *Electrónica e Telecomunicações*, vol. 4 (4), Universidade de Aveiro, Portugal, p. 467-477.
5. Carter, B., et al. (2002) "Mechanical Design and Modeling of an Omni-directional RoboCup Player", *RoboCup-2001*, A. Birk, et al (edrs), Springer Verlag.
6. Kopetz, H. (1997) Real-Time Systems Design Principles for Distributed Embedded Applications, Kluwer.
7. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos (2005) Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup Symposium: Papers and Team Description Papers* [CD], to appear in I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, LNAI, Springer, 2006.
8. Santos, F., L. Almeida, P. Pedreiras, L. Seabra Lopes, T. Facchinetti (2004) An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among, Mobile Autonomous Agents, *Proc. WACERTS'2004, Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems* (in conjunction with RTSS 2004), Lisboa, Portugal.
9. Vieira, J.M.N., S.I. Lopes, C.C. Bastos and P.N. Fonseca (2005) "Sistema de Localização Mútua para Robots Utilizando Ultra-Sons", *JETC05*, ISEL, Lisboa, Portugal.



Apêndice B

ROBOCUP MIDDLE SIZE LEAGUE REFEREE BOX

ROBOCUP MIDDLE SIZE LEAGUE REFEREE BOX

Gustavo A. Corrente
IEETA – Universidade de Aveiro
Campus Univ. de Santiago 3810-193 Aveiro PORTUGAL
gustavo@ieeta.pt

ABSTRACT

This paper presents a new architecture for the Referee Box of the RoboCup Middle Size League competition. The Referee Box is the software that helps the human referee controlling the robotic soccer game. This architecture is based on a distributed and parallel paradigm. Three different modules were developed: server, control and viewer.

KEYWORDS

Distributed and Parallel Systems, Robot-Human Interaction, RoboCup Middle Size League.

1. INTRODUCTION

RoboCup [1] is a non-profit organization that promotes developments in the fields of Robotic and Artificial Intelligence [2]. This objective is reached by developing robots to play soccer. RoboCup is composed of several leagues like small size, middle size, simulation 2D, simulation 3D, and humanoid. The ultimate goal of RoboCup is: "by the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team". Robots of the Middle Size League (MSL) are around 80cm height and 50cm width. The game is played in a color coded environment to make lees difficult the image processing. The soccer field is green with 12m length and 8m width, white lines, and two goals (blue and yellow) [3].

Until 2004 the human referee communicated its decisions to teams using only speech. Now each game has the following actors: human referee team (main referee, assistant referee and referee box operator), base-stations and robotic players. Base-station is a computer that stays out of field, connected to the Referee Box, used for communicating the referee's decision to robotic players, received from referee box. Each team has their own base-station.

During MSL games, the human referee team must communicate their decisions to the competing teams, the Referee Box makes that bridge. The Referee Box is the software, used to aid the referee team controlling the game and manages the communication with the teams.

Currently the procedure during a game is the following: before starting the game every base-station must establish a connection to the referee box. The main referee communicates his decision to the referee box operator using speech and a whistle; the operator inputs the decision into the referee box software; the decision is transmitted to all connected base-stations and each one must communicate the decisions to their own robots.

The current referee box software [4] is a single process without threads that merges the game management, the graphical interface and deals with the communication with the team's base-stations. This software has architecture and implementation problems. The referee box waits for a pre-defined number of team connections and only at this moment starts up; the recovery, if one team looses connection, is made by writing the actual state in a file; doesn't support a bidirectional communication between referee box and teams.

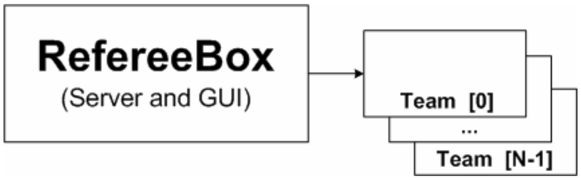
In this paper the proposed architecture is fully based on a distributed and parallel paradigm. This architecture is composed by three modules: Server, Control and Viewer. The Server Module manages the game and deals with the communication from teams. The Control Module is responsible for manage the Server Module through XML messages. The Viewer Module provides information of the game state (i.e.

result, competing teams, time and score) for the public that see the game, like in a real soccer game. It is possible to have more than one module of the same type at the same time (Control and Viewer Modules). For example, we can split the game refereeing and use two different control modules, one for controlling the normal game flux and other to deal with substitutions. This also supports functionalities like managing substitutions, cards and goals.

2. THE CURRENT REFEREE BOX ARCHITECTURE

The current architecture of the referee box is a centralized solution *(see Figure 1)*, mixing referee box management, graphical interface and communication with teams. In this architecture we can find disadvantages in implementation, usability.

Figure 1 : Current architecture of referee box

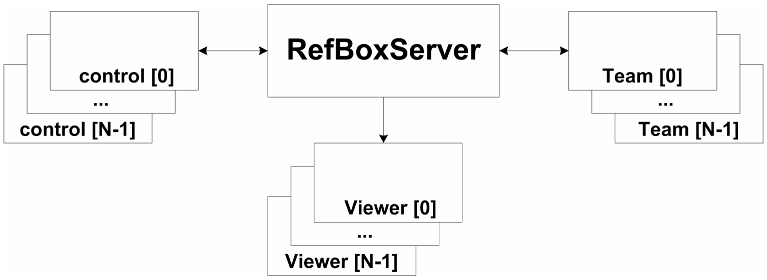


The referee box waits for a fixed number of teams and only starts up when all teams are connected. All referee decisions are converted to a character protocol and transmitted to connected teams. This character protocol is poor because it is based on a single character message used for the communication between referee box and teams (only this direction). The character protocol can not send information like time, cards or substitutions. Another other problem is related with disconnections of the teams. When this happen the graphical interface is frozen and the recovery is made by saving the referee state in a file.

3. THE PROPOSED REFEREE BOX ARCHITECTURE

The proposed architecture has been designed and implemented to provide new functions and support future function developments [5]. This objective was reached by three modules: RefBoxServer (Server Module), Control and Viewer *(see Figure 2)*.

Figure 2 : Proposed architecture of referee box

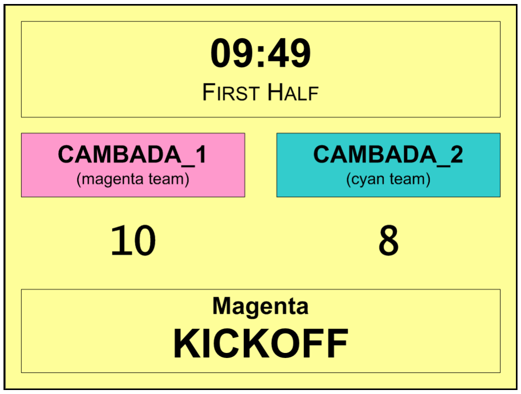


The objective of Control Module is managing the Server Module by event interaction. The referee interacts with the Control Module through a graphical interface, as result of that interaction XML messages are sent to the Server Module. This use XML messages represent events like: start/stop all robots, fouls and register goals cards and substitutions. The Control receives from the Server the game state information. It is possible to use two (or more) different implementation of Control, for example: one for controlling the game (fouls, goal and cards) and other to deal with the substitutions.

The Viewer Module is in charge of displaying the game state like an electronic placard used in a real soccer game. This module is connected to Server and receives periodically information like: name of the competing team, time, and score and current foul *(see Figure 3).*

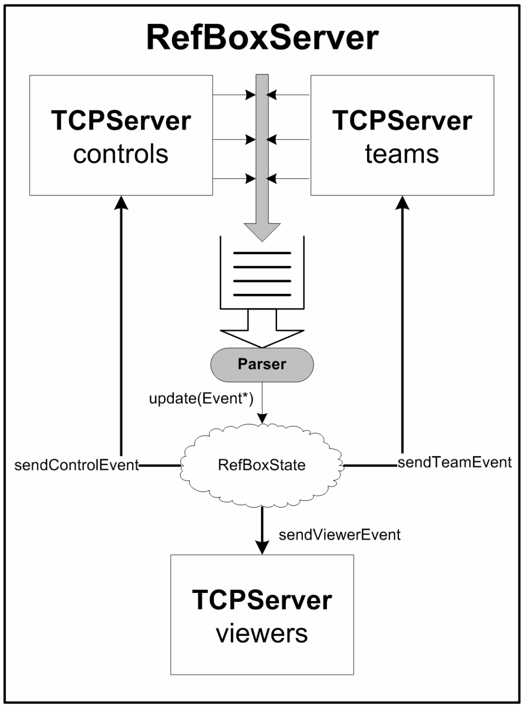
The Server Module is responsible for broadcasting decision events like: cards, goals, time or substitutions to teams and viewer. This module is also responsible for managing all game state information. The Server Module uses three TCPServer objects for dealing with teams, controls and viewers. The communications between Server/Control and Server/Team are bidirectional and unidirectional in case of connection of Server/Viewer. The Server/Control connection is used for controlling the game state. Control sends XML events to change the game state in the Server. Each XML message, received by TCPServer (for control purpose), is tagged (by TCPServer) and placed in a queue.

Figure 3 : Referee box Viewer graphical interface



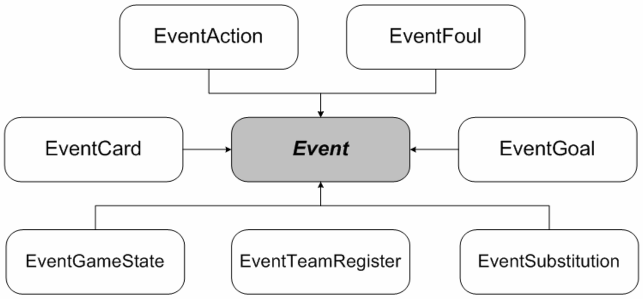
The Server Module is waiting for message from the queue. After getting a message the Parser module is used to convert XML messages to event objects. These objects are used to update the RefBoxState (game state) *(see Figure 4)*. The RefBoxState is a data structure that supports the storage of game, referee and team information.

Figure 4 : Referee box server


RefBoxServer
TCPServer controls
TCPServer teams
Parser
update(Event*)
sendControlEvent
RefBoxState
sendTeamEvent
sendViewerEvent
TCPServer viewers

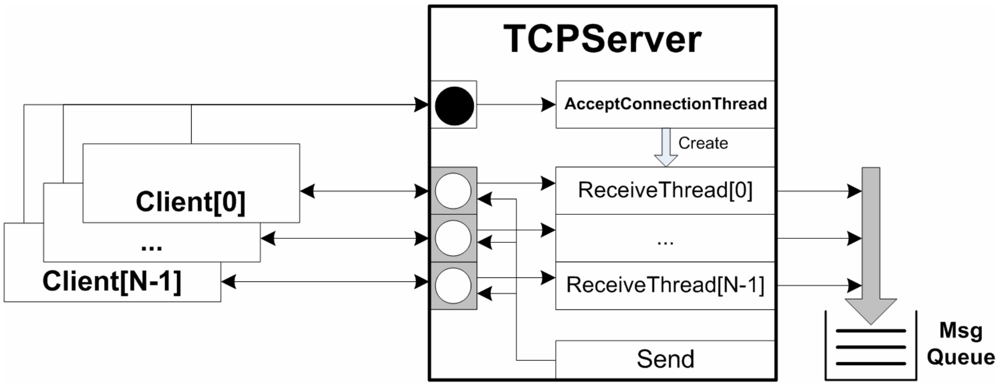
The RefBoxState is updated with events, in *Figure 5* we can see how they are organized. All events are derived from an abstract base event called Event. We can configure each event to be forwardable or not and the method used to write it self. The write method can be configured in two modes: character mode (like the current referee box protocol) or XML protocol.

Figure 5 : Events diagram


EventAction
EventFoul
EventCard
Event
EventGoal
EventGameState
EventTeamRegister
EventSubstitution

TCPServer has one well-known port, (called connection channel) for accepting connections to the server. When one client connects to that port the server assigns at another port, for communication channel and creates a thread. For each new port a new thread is created for receiving data from socket [6]. When a message is received at a port, the correspondents thread tags the message and places it in the message queue. This tag is the port number of TCPServer. When the Send function is called for dispatching a message, it sends the information to all connected clients.

Figure 6 : TCPServer architecture



4. CONCLUSION AND FUTURE WORK

The proposed architecture is modular and distributed. These two features of the architecture causes: easy development of new modules for implementing new functions to the system. New functions have been added like substitutions and time report to this system. It also adds a XML protocol to make the communication between modules easier and flexible. With this approach the game management is more easy and natural. In a real game the communication between the main referee and the referee box operator is difficult because the noise on the field. This problem can be solved by the main referee control: player's motion, goals, cards and game fouls; and the referee box operator can control the substitution.

A possible limitation of the architecture is the fact that the interoperability between modules is made by a XML protocol.

To future work, two new modules are in discussion. One is a new Control Module running in a Pocket PC for the main referee. The other module is a voice control to the referee box

ACKNOWLEDGEMENT

The author wishes to acknowledge Nuno Lau, for the definition of this architecture and preparation of this working paper.

REFERENCES

[1] RoboCup home page, http://robocup.org.

[2] Hiroaki Kitano et al, 1997. RoboCup: The Robot World Cup Initiative. *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*. Marina Del Rey, Ca USA, pp. 340-347.

[3] RoboCup Middle Size League Rule and Regulations home page, http://www.er.ams.eng.osaka-u.ac.jp/robocup-mid/index.cgi?page=Rules+and+Regulations.

[4] Middle Size League home page, http://sourceforge.net/projects/msl-refbox.

[5] Bertrand Meyer, 1997. *Object-Oriented Software Construction*. Prentice Hall, Upper Saddle River (NJ), USA.

[6] Andrew S. Tanenbaum et al, 2002. *Distributed Systems, Principles and Paradigm*. Prentice Hall, New Jersey, USA.

Apêndice C

An Omnidirectional Vision System for Soccer Robots

An Omnidirectional Vision System for Soccer Robots

António J. R. Neves[⋆], Gustavo A. Corrente and Armando J. Pinho

Dept. de Electrónica e Telecomunicações / IEETA
Universidade de Aveiro, 3810–193 Aveiro, Portugal
an@ua.pt, gustavo@ua.pt, ap@ua.pt

Abstract. This paper describes a complete and efficient vision system developed for the robotic soccer team of the University of Aveiro, CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture). The system consists on a firewire camera mounted vertically on the top of the robots. A hyperbolic mirror placed above the camera reflects the 360 degrees of the field around the robot. The omnidirectional system is used to find the ball, the goals, detect the presence of obstacles and the white lines, used by our localization algorithm. In this paper we present a set of algorithms to extract efficiently the color information of the acquired images and, in a second phase, extract the information of all objects of interest. Our vision system architecture uses a distributed paradigm where the main tasks, namely image acquisition, color extraction, object detection and image visualization, are separated in several processes that can run at the same time. We developed an efficient color extraction algorithm based on lookup tables and a radial model for object detection. Our participation in the last national robotic contest, ROBOTICA 2007, where we have obtained the first place in the Medium Size League of robotic soccer, shows the effectiveness of our algorithms. Moreover, our experiments show that the system is fast and accurate having a maximum processing time independently of the robot position and the number of objects found in the field.

Keywords: Robotics; robotic soccer; computer vision; object recognition; omnidirectional vision; color classification.

1 Introduction

After Garry Kasparov was defeated by IBM's Deep Blue Supercomputer in May 1997, forty years of challenge in the artificial intelligence (AI) community came to a successful conclusion. But it also was clear that a new challenge had to be found.

”By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rules of the FIFA, against the winner of the most recent World Cup.” This is how the ultimate goal was stated by the RoboCup Initiative, founded in 1997, with the aim to foster the development of artificial intelligence and related field by providing a standard problem: robots that play soccer.

It will take decades of efforts, if not centuries, to accomplish this goal. It is not feasible, with the current technologies, to reach this goal in any near term. However,

---

[⋆] This work was supported in part by the Fundação para a Ciência e a Tecnologia (FCT).

this goal can easily create a series of well directed subgoals. The first subgoal to be accomplished in RoboCup is to build real and simulated robot soccer teams which play reasonably well with modified rules. Even to accomplish this goal will undoubtedly generate technologies with impact on broad range of industries.

One problem domain in RoboCup is the field of Computer Vision. Its task is to provide basic information that is needed to calculate the behavior of the robots. Especially omnidirectional vision systems have become interesting in the last years, allowing a robot to see in all directions at the same time without moving itself or its camera [13, 12, 10, 11]. Omnidirectional vision is the method used by most teams in the Middle Size League.

The main goal of this paper is to present an efficient vision system for processing the video acquired by an omnidirectional camera. The system finds the white lines of the playing field, the ball, goals and obstacles. The lines of the playing field are needed because knowing the placement of the playing field from the robot's point of view is equal to know the position and orientation of the robot.

For finding the goals, the ball and the obstacles, lines are stretched out radially from the center of the image and, if some defined number of pixels of the respective colors are found, the system saves that position associated to the respective color. For finding the white lines, color transitions from green to white are searched for.

For color classification, the first step of our system, a lookup table (LUT) is used. Our system is prepared to acquire images in RGB 24-bit, YUV 4:2:2 or YUV 4:1:1 format, being necessary only to choose the appropriated LUT. We use the HSV color space for color calibration and classification due to its special characteristics [1].

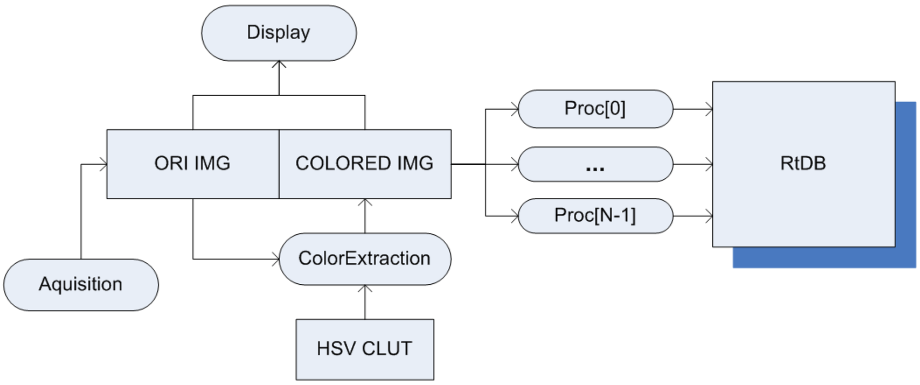


Fig. 1. The architecture of our vision system. It is based on a multi-process system being each process responsible for a specific task.

This paper is organized as follows. In Section 2 we describe the design of our robots. Section 3 presents our vision system architecture, explaining the several modules developed and how they are connected. In Section 4 we present the algorithms used to collect the color information of the image using radial search lines. In Section 5 we describe how the object features are extracted. Finally, in Section 6, we draw some conclusions and propose future work.

2 Robot overview

CAMBADA players were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter. The mechanical structure of the players is layered and modular. The components in the lower layer are the motors, wheels, batteries and an electromechanical kicker. The second layer contains the control electronics. The third layer contains a computer. The players are capable of holonomic motion, based on three omni-directional roller wheels [2].

The vision system consists on a firewire camera mounted vertically on the top of the robots. A hyperbolic mirror placed above the camera reflects the 360 degrees of the field around the robot. This is the main sensor of the robot and it is used to find the ball, the goals, detect the presence of obstacles and the white lines.

The robots computing system architecture follows the fine-grain distributed model [7] where most of the elementary functions are encapsulated in small microcontroller-based nodes, connected through a network. A PC-based node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. Firewire. For this purpose, Controller Area Network (CAN) has been chosen [3]. The communication among robots uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment.

The software system in each player is distributed among the various computational units. High level functions run on the computer, in Linux operating system with RTAI (Real-Time Application Interface). Low level functions run partly on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [4–6] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

3 Vision system architecture

A modular multi-process architecture was adopted for our vision system (see Fig. 1).

When a new frame is ready to download, one process is automatically triggered and the frame is placed in a shared memory buffer. After that, another process analyzes the acquired image for color classification, creating a new image with “color labels” (an 8 bpp image). This image is also placed in the shared image buffer, which is afterward analyzed by the object detection processes, generically designated by $Proc[x]$, $x = 1, 2, \dots N$. These applications are encapsulated in separate Linux processes. Once started, each process gets a pointer to the most recent image frame available and starts tracking the respective object. Once finished, the resulting information (e.g. object detected or not and position) is placed in the real-time database. This database may be accessed by any other processes in the system, particularly for world state update.

The activation of the distinct image-processing activities is carried out by a process manager. Scheduling of vision related processes relies on the real-time features of the Linux kernel, namely the FIFO scheduler and priorities. At this level, Linux executes each process to completion, unless the process blocks or is preempted by other process with higher real-time priority. This ensures that the processes are executed strictly according to their priority with full preemption.

4 Color extraction

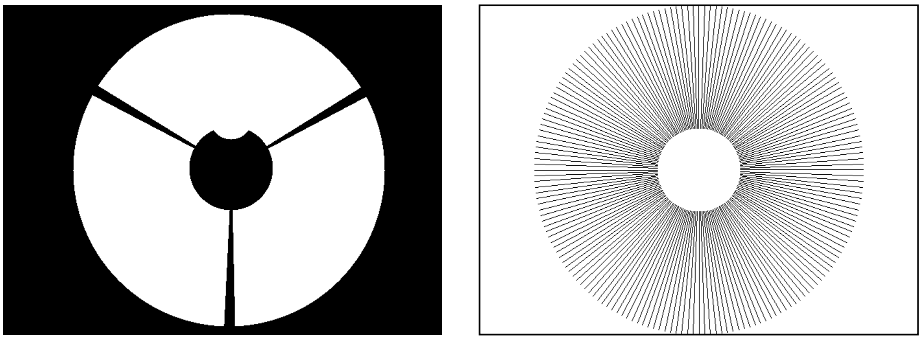


Fig. 2. On the left, an example of a robot mask. White points represent the area that will be processed. On the right, the position of the radial search lines.

Image analysis in the RoboCup domain is simplified, since objects are color coded. Black robots play with an orange ball on a green field that has yellow and blue goals and white lines. Thus, a pixel's color is a strong hint for object segmentation. We exploit this fact by defining color classes, using a look-up table (LUT) for fast color classification. The table consists of 16777216 entries (2^{24}, 8 bits for red, 8 bits for green and 8 bits for blue), each 8 bits wide, occupying 16 MB in total. If another color space is used, the table size is the same, changing only the "meaning" of each component. Each bit expresses whether the color is within the corresponding class or not. This means that a certain color can be assigned to several classes at the same time. To classify a pixel, we first read the pixel's color and then use the color as an index into the table. The value (8 bits) read from the table will be called "color mask" of the pixel.

The color calibration is done in HSV (Hue, Saturation and Value) color space due to its special characteristics. In our system, the image is acquired in RGB or YUV format and then is converted to HSV using an appropriate conversion routine.

There are certain regions in the received image that have to be excluded from analysis. One of them is the part in the image that reflects the robot itself. Other regions are the sticks that hold the mirror and the areas outside the mirror. For that, we have an image with this configuration that is used by our software. An example is presented in Fig. 2. The white pixels are the area that will be processed, black pixels will not. With this approach we can reduce the time spent in the conversion and searching phases and we eliminate the problem of finding erroneous objects in that areas.

To extract the color information of the image we use radial search lines to analyze the color information instead of processing all the image. This approach has two main advantages. First, that of accelerating the process due to the fact that we only process about 30% of the valid pixels. Second, the use of omnidirectional vision difficults the detection of the objects using, for example, their bounding box. In this case, it is more desirable to use the distance and angle. The proposed approach has a processing time almost constant, independently of the information around the robot, being a desirable property in Real-Time Systems. This is due to the fact that the system processes almost the same number of pixels in each frame.

A radial search line is a line that starts in the center of the robot with some angle and ends in the limit of the image (see the image on the right of Fig. 2). They are constructed based on the Bresenham line algorithm [8, 9]. For each search line, we iterate through its pixels to search for two things: transitions between two colors and areas with specific colors.

We developed an algorithm to detect areas of a specific color which eliminates the possible noise that could appear in the image. Each time that a pixel is found with a color of interest, we analyze the pixels that follows (a predefined number) and if we don't find more pixels of that color we "forget" the pixel found and continue. When we find a predefined number of pixels with that color, we consider that the search line has this color.

To accelerate the process of calculating the position of the objects, we put the color information found, in each search line, into a list of colors. We are interested in the first pixel (in the respective search line) where the color was found and the number of pixels with that color found in the search line. Then, using the previous information, we separate the information of each color into sets that we named blobs (see Fig. 3). For each blob some useful information is calculated that will help in the detection of each object:

- average distance to the robot;
- mass center;
- angular width;
- number of pixels;
- number of green pixels between blob and the robot;
- number of pixels after blob.

The algorithm to search for the transitions between green pixels and white pixels is described as follows. If a non green pixel is found, we will look for a small window in the "future" and count the number of non green pixels and the number of white pixels. Next, we look for a small window in the "past" and a small window in the future and count the number of green pixels. If these values are greater than a predefined threshold, we consider this point as a transition. This algorithm is illustrated in Fig. 4.

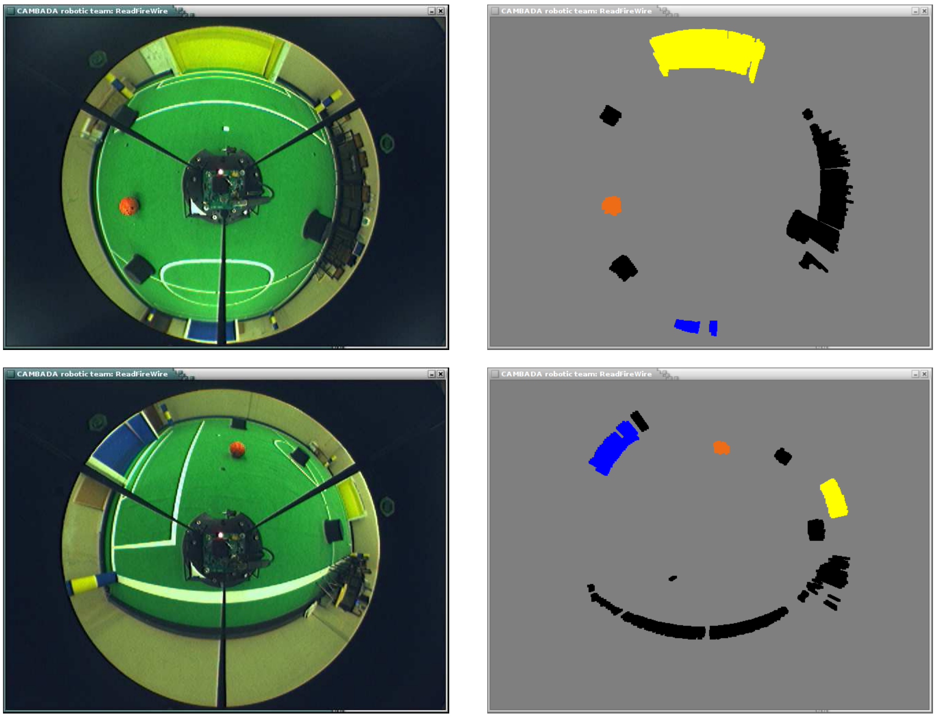
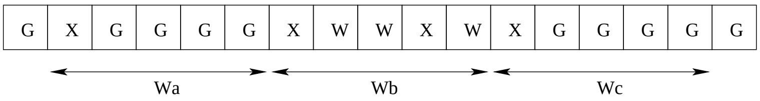

CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire

Fig. 3. An example of the blobs found in two images. On the left, the original images. On the right, the blobs found. For each blob, we calculate useful information that is used later to calculate the position of each object.



G	X	G	G	G	G	X	W	W	X	W	X	G	G	G	G	G

Fig. 4. An example of a transition. “G” means green pixels, “W” means white pixels and “X” means pixels with a color different from green or white.

5 Object detection

The objects of interest that are present in the RoboCup environment are: a ball, two goals, obstacles (other robots) and the green field with white lines. Currently, our system detects efficiently all these objects with a set of simple algorithms that, using the color information collected by the radial search lines, calculate the object position and / or their limits in an angular representation (distance and angle).

The transition points detected in the color extraction phase are used for the robot localization. All the points detected are sent to the Real-time Database, afterward used by the localization process.

To detect the ball, we use the following algorithm:

1. Separate the orange information into blobs.
2. For each blob, calculate the information described in the previous section.
3. Sort the orange blobs that have some green pixels before or after the blob by descending order, using their number of orange pixels as measure.
4. Choose the first blob as candidate. The position of the ball is the mass center of the blob.

Regarding the goals, we are interested in three points: the center, the right post and the left post. To do that, we use the following algorithm:

1. Ignore the information related to radial search lines which have both blue and yellow information (they correspond to the land marks).
2. Separate the valid blue and yellow information into blobs.
3. Calculate the information for each blob.
4. Sort the yellow and the blue blobs that have some green pixels before the blob by descending order, using their angular width as measure.
5. Choose the first blob as candidate for each goal. Their position is given by the distance of the blob relatively to the robot.
6. The right post and the left post is given by the position of the goal and the angular width of the blob.

Another important information regarding the goals, is the best point to shoot. To calculate it, we split the blob chosen into several slices, and choose the one with most pixels blue or yellow. The best point to shoot is the mass center of the slice chosen.

To calculate the position of the obstacles around the robot, we use the following algorithm:

1. Separate the orange information into blobs.
2. If the angular width of one blob is greater than 10 degrees, we split the blob into smaller blobs, in order to obtain better information about obstacles.
3. Calculate the information for each blob.
4. The position of the obstacle is given by the distance of the blob relatively to the robot. We are also interested in the limits of the obstacle and to obtain that we use the angular width of the blob.

In Fig. 5 we present some examples of acquired images and their correspondent segmented images. As we can see, the objects are correctly detected (see the marks in images on the right).

6 Final remarks

This paper presents the omnidirectional vision system that has been developed for the CAMBADA team. We present several algorithms for image acquisition and processing. The experiments already made and the last results obtained in the ROBOTICA 2007 competition prove the effectiveness of our system regarding the object detection and robot self-localization.

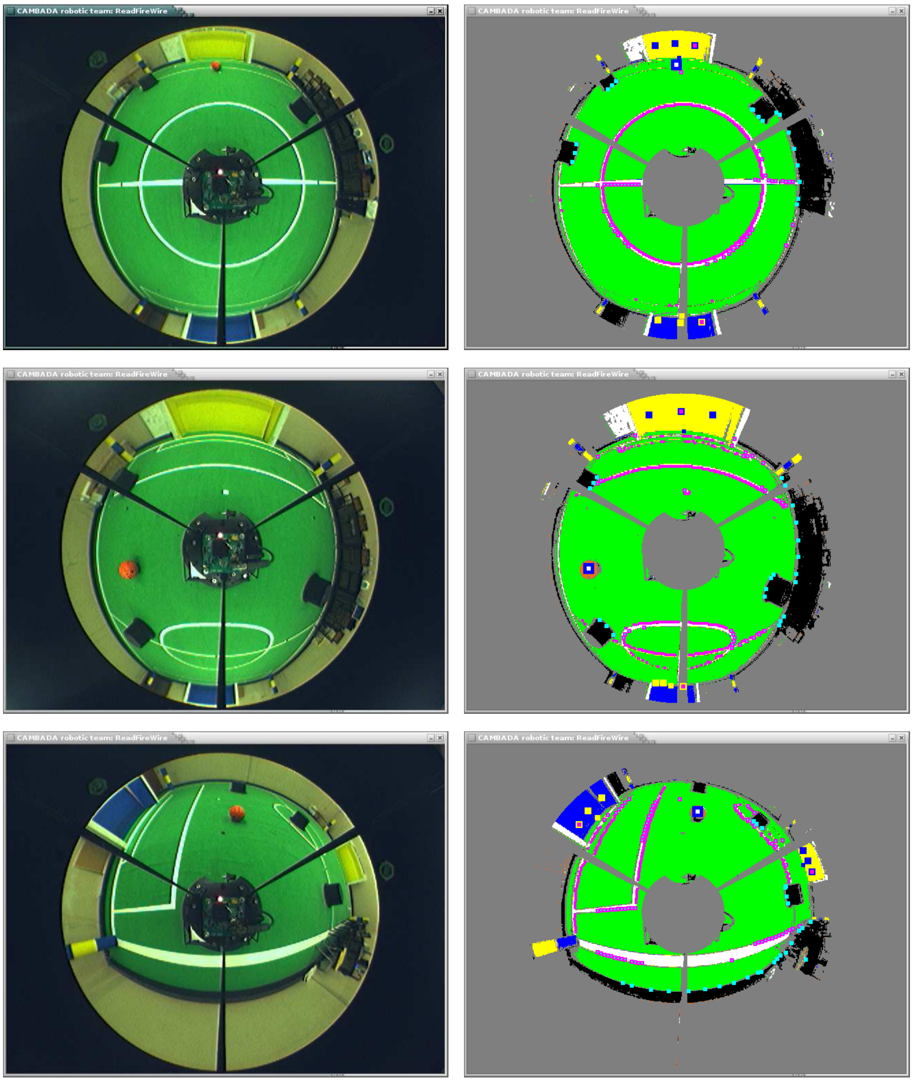

CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire

Fig. 5. On the left, examples of original images. On the right, the corresponding processed images. Marks in the blue and yellow goals mean the position of the goal (center) and the possible points to shoot. The mark over the ball points to the mass center. The several marks near the white lines (magenta) are the position of the white lines. The cyan marks are the position of the obstacles.

The objects in RoboCup are color coded. Therefore, our system defines different color classes corresponding to the objects. The 24 bit pixel color is used as an index to a 16 MBytes lookup table which contains the classification of each possible color in a 8 bit entry. Each bit specifies whether that color lays within the corresponding color class.

The processing system is divided in two phases: color extraction, using radial search lines, and object detection, using specific algorithms. The objects involved are: a ball, two goals, obstacles and white lines. The processing time and the accuracy obtained in the object detection confirms the effectiveness of our system.

As future work, we are developing new algorithms for camera and color calibration, in particular autonomous algorithms. Moreover, we are improving the presented algorithms in order to use the shape of the objects instead of using only the color information to improve the object recognition.

References

1. Pedro M. R. Caleiro, António J. R. Neves and Armando J. Pinho, Color-spaces and color segmentation for real-time object recognition in robotic applications, *Revista do DETUA*, Vol. 4, N. 8, Junho 2007, pp. 940-945.
2. Carter, B., *EST LA.*: Mechanical Design and Modeling of an Omni-directional RoboCup Player, *RoboCup-2001*, A. Birk, *et al* (edrs), Springer Verlag.
3. Almeida, L., P. Pedreiras and J.A. Fonseca: FTT-CAN: Why and How, *IEEE Trans. Industrial Electronics*, 2002.
4. Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes: Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences – ISCIS 2004: 19th International Symposium, Proceedings*, Lecture Notes in Computer Science, Vol. 3280, p. 876-886.
5. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos: Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup Symposium: Papers and Team Description Papers, RoboCup-2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2006.
6. Santos, F., L. Almeida, P. Pedreiras, L. Seabra Lopes, T. Facchinetti: An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among, Mobile Autonomous Agents, *Proc. WACERTS'2004, Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems (in conjunction with RTSS 2004)*, Lisboa, Portugal.
7. Kopets H.: Real-Time Systems Design Principles for Distributed Embedded Applications, Kluwer.
8. Bresenham, J. E.: A linear algorithm for incremental digital display of circular arcs, *CA CM*, 20(2), pp. 100-106, 1977.
9. Bresenham, J. E.: Algorithm for computer control of a digital plotter, *IBM Systems J.*, 4(1), pp.25-30, 1965.
10. P. Heinemann *et al.*: Fast and Accurate Environment Modelling using Omnidirectional Vision, *Dynamic Perception*, pp. 9-14, Infix, 2004.
11. P. Heinemann *et al.*: Tracking Dynamic Objects in a RoboCup Environment - The Attempto Tübingen Robot Soccer Team, *RoboCup-2003: Robot Soccer World Cup VII*, LNCS, Volume 3020, Springer, 2003.
12. Jose Gaspar, Niall Winters, Etienne Grossmann, Jose Santos-Victor: Toward Robot Perception using Omnidirectional Vision, *In Innovations in Machine Intelligence and Robot Perception*, Springer-Verlag, 2004.
13. Jan Hoffmann *et al.*: A vision based system for goal-directed obstacle avoidance, *RoboCup-2004: Robot Soccer World Cup VIII*, LNCS, Springer, 2004.

Apêndice D

Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots

Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots

Frederico Santos[1], Gustavo Currente[2], Luís Almeida[2],
Nuno Lau[2] and Luís Seabra Lopes[2]

[1] DEE, Instituto Superior de Engenharia de Coimbra,
Rua Pedro Nunes, 3030-199 Coimbra, Portugal
`fred@isec.pt`
[2] IEETA - DETI, Universidade de Aveiro,
Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{`gustavo, lda, nunolau, lsl`}`@ua.pt`

Abstract. Interest on using mobile autonomous agents has been growing, recently, due to their capacity to cooperate for diverse purposes, from rescue to demining and security. However, such cooperation requires the exchange of state data that is time sensitive and thus, applications should be aware of data temporal coherency. This paper describes the communication and coordination architecture of the agents that constitute the CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) robotic soccer team developed at the University of Aveiro, Portugal. This architecture is built around a partially replicated real-time database refreshed in the background, transparently to the higher software layers. The paper presents the communication mechanisms that were devised to support the real-time database management and focuses on the self-configuration of the protocol, according to the current number of active team members.

1 Introduction

Coordinating several autonomous mobile robotic agents in order to achieve a common goal is an active topic of research [3]. This problem can be found in many robotic applications, either for military or civil purposes, such as search and rescue in catastrophic situations, demining or maneuvers in contaminated areas.

The technical problem of building an infrastructure to support the perception integration for a team of robots and subsequent coordinated action is common to the above applications. One recent initiative to promote research in this field is RoboCup [5] where several autonomous robots have to play football together as a team, to beat the opponent. We believe that researching ways to solve the perception integration problem in RoboCup is also very relevant to real-world applications.

Currently, the requirements posed on such teams of autonomous robotic agents have evolved in two directions. On one hand, robots must move faster and with accurate trajectories to close the gap with the dynamics of the processes they interact with, e.g., a ball can move very fast. On the other hand, robots must interact more in order to develop coordinated actions more efficiently, e.g., only the robot closer to the ball should try to get it while other robots should move to appropriate positions. The former requirement demands for tight closed-loop motion control while the latter demands for an appropriate communication system that allows building a global information base to support cooperation.

In this paper we describe the communication and coordination architecture of the robotic agents that constitute the CAMBADA middle-size robotic soccer team of the University of Aveiro, Portugal, which is well suited to support the requirements expressed above. The software architecture is based on a real-time database in which the state values of other agents are updated transparently to the higher software layers, using an adequate communication protocol. This paper focuses on such protocol that dynamically adapts to the conditions of the communication channel and to the current number of active agents in the team. Particularly, some results are shown that illustrate the latter self-configuration capability.

2 Computing/Communications Architecture

The computing architecture of the robotic agents is layered with two levels as illustrated in Fig. 1. The higher level is built around a main processing unit that handles both the external communication with other agents as well as the local vision system. A distributed low-level sensing/actuating system handles the robot attitude (holonomic motion control), odometry, kicking and power monitoring. The latter one is out of the scope of this paper.

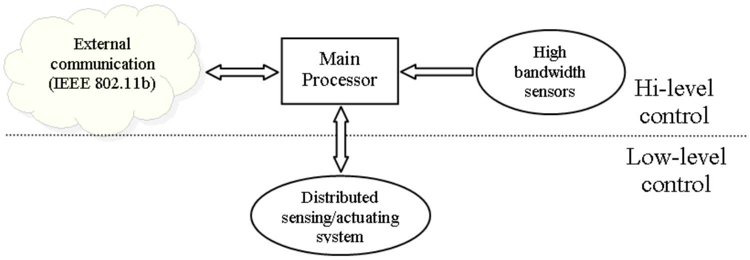

External communication (IEEE 802.11b)
Main Processor
High bandwidth sensors
Hi-level control
Low-level control
Distributed sensing/actuating system

Fig. 1. CAMBADA robotic architecture

The main processing unit is currently implemented on a laptop that delivers sufficient computing power while offering standard interfaces to connect the other systems, namely USB, FireWire and WiFi. The wireless interface is either built-in or added as a PCMCIA card. The laptop runs the Linux operating system with the timeliness support necessary for time-stamping, periodic transmissions and task temporal synchronization provided by a specially developed user-level real-time scheduler, the Pman – Process Manager [1]. This approach provides a sufficient timeliness support for soft real-time applications, such as multiple robot coordination, and allows profiting from the better development support provided by general purpose operating systems [2].

The agents that constitute the team communicate with each other by means of an IEEE 802.11b wireless network as depicted in Fig. 2. The communication is managed, i.e., using a base station, and it is constrained to using a single channel, shared by, at least, both teams in each game. In order to improve the timeliness of the communications, our team uses a further transmission control protocol that minimizes collisions of transmissions within the team. Each robot regularly broadcasts its own data while the remaining ones receive such data and update their local structures. Beyond the robotic agents, there is also a coaching and monitoring station connected to the team that allows following the evolution of the robots status on-line and issuing high level team coordination commands.

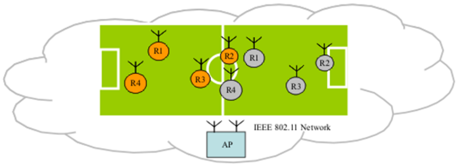


Fig. 2. Global communications architecture

3 RTBD - The Real-Time Database

Similarly to other teams [4, 6], our team software architecture emphasizes cooperative sensing as a key capability to support the behavioral and decision-making processes in the robotic players. A common technique to achieve cooperative sensing is by means of a *blackboard* [7, 8], which is a database where each agent publishes the information that is generated internally and that maybe requested, by others. However, typical implementations of this technique seldom account for the temporal validity (coherence) of the contained information with adequate accuracy. This is a problem when robots move fast because their state information degrades faster, too. Without adequate refreshing, the data in a blackboard may easily lose temporal validity thus becoming too old to be useful. Another problem of typical implementations is that they are based on the client-server model and thus, when a robot needs a datum, it has to communicate with the server holding the blackboard, introducing an undesirable delay. To avoid this delay, we use two features: firstly, the dissemination of the local state data is carried out using multicast, according to the producer-consumer cooperation model, secondly, we replicate the blackboard according to the *distributed shared memory* model [9]. In this model, each node has local access to all the process state variables that it requires. Those variables that are remote have a local image that is updated automatically in the background by the communication system (Fig. 3).

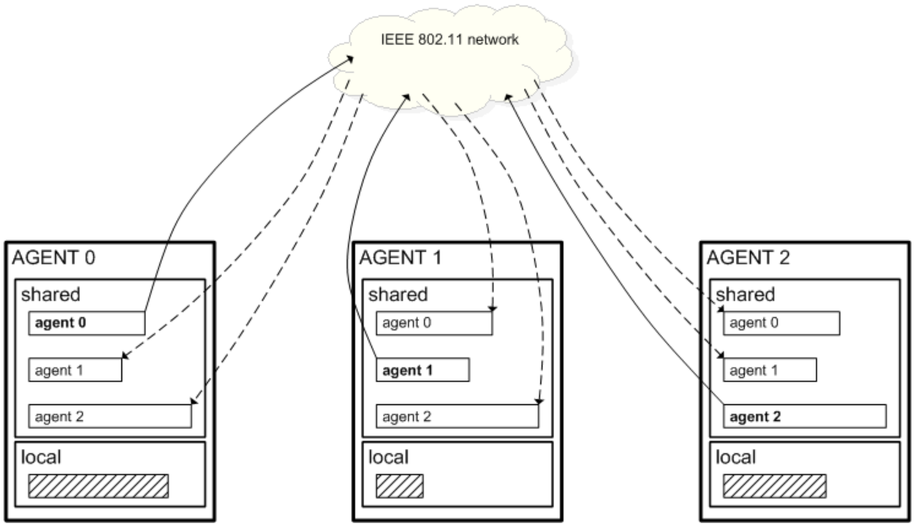


Fig. 3. Each agent broadcasts periodically its subset state data that might be required by other agents

We call this replicated blackboard the Real-Time DataBase (RTDB), similarly to the concept presented in [10], which holds the state data of each agent together with local images of the relevant state data of the other team members. A specialized communication system triggers the required transactions at an adequate rate to guarantee the freshness of the data.

4 Communication Among Agents

As referred in section 2, agents communicate using an IEEE 802.11 network, sharing a single channel with the opposing team and using managed communication (through the access point). This raises several difficulties because the access to the channel cannot be controlled [11] and the available bandwidth is roughly divided by the two teams.

Therefore, the only alternative left for each team is to adapt to the current channel conditions and reduce access collisions among team members. This is achieved using a dynamic adaptive TDMA transmission control, with a predefined round period called *team update period (T_{tup})* that sets the responsiveness of the global communication. Within such round, there is one single slot allocated to each running team member so that all slots in the round are separated as much as possible (Fig. 4). This allows calculating the target inter-slot period T_{xwin} as T_{tup}/N, where N is the number of running agents. The transmissions generated by each running agent are scheduled within the communication process, according to the production periods specified in the RTDB records. Currently a rate-monotonic scheduler is used. When the respective TDMA slot comes, all currently scheduled transmissions are piggybacked on one single 802.11 frame and sent to the channel. The required synchronization is based on the reception of the frames sent by the other agents during T_{tup}. With the reception instants of those frames and the target inter-slot period T_{xwin} it is possible to generate the next transmission instant. If these delays affect all TDMA frames in a round, then the whole round is delayed from then on, thus its adaptive nature. Fig. 5 shows a TDMA round indicating the slots allocated to each agent and the adaptation of the round duration. The adaptive TDMA protocol was first proposed by the authors in [12].

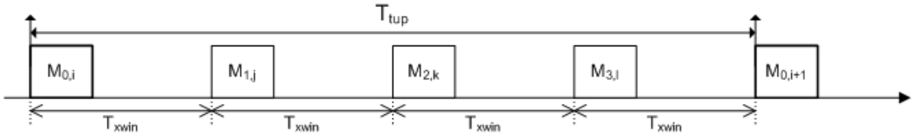


Fig. 4. TDMA round

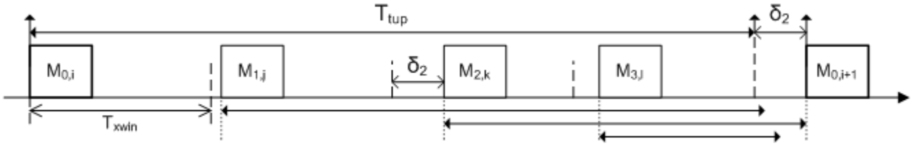


Fig. 5. Adaptive TDMA round

When a robot transmits at time t_{now} it sets its own transmission instant $t_{next} = t_{now} + T_{tup}$, i.e. one round after. However, it continues monitoring the arrival of the frames from the other robots. When the frame from robot k arrives, the delay δ_k of the effective reception instant with respect to the expected instant is calculated. If this delay is within a validity window $[0, \Delta]$, with Δ being a global configuration parameter, the next transmission instant is delayed according to the longest such delay among the frames received in one round (Fig. 5), i.e.,

$$t_{next} = t_{now} + T_{tup} + max_k(\delta_k)$$

On the other hand, if the reception instant is outside that validity window, or the frame is not received, then δ_k is set to 0 and does not contribute to update t_{next}.

The practical effect of the protocol is that the transmission instant of a frame in each round may be delayed up to Δ with respect to the predefined round period T_{tup}. Therefore, the effective round period will vary between T_{tup} and $T_{tup} + \Delta$. When a robot does not receive any frame in a round within the respective validity windows, it updates t_{next} using a robot specific configuration parameter β_k in the following way

$$t_{next} = t_{now} + T_{tup} + \beta_k \quad with \quad 0 \leq \beta_k \leq \Delta$$

This is used to prevent a possible situation in which the robots could all remain transmitting but unsynchronized, i.e. outside the validity windows of each other, and with the same period T_{tup}. By imposing different periods in this situation we force the robots to resynchronize within a limited number of rounds because the transmissions will eventually fall within the validity windows of each other.

One of the limitations of the adaptive TDMA protocol as proposed in [12] is that the number of team members was fixed, even if the agents were not active, causing the use of T_{xwin} values smaller than needed. Notice that a smaller T_{xwin} increases the probability of collisions in the team. Therefore, a self-configuration capability was added to the protocol, to cope with variable number of team members. This is the specific mechanism proposed in this paper, which supports the dynamic insertion / removal of agents in the protocol. Currently, the T_{tup} period is still constant but it is divided by the number of running agents at each instant, maximizing the inter-slot separation between agents T_{xwin} at each moment.

However, the number of active team members is a global variable that must be consistent so that the TDMA round is divided in the same number of slots in all agents. To support the synchronous adaptation of the current number of active team members a membership vector was added to the frame transmitted by each agent in each round, containing its perception of the team status.

When a new agent arrives it starts to transmit its periodic information in an unsynchronized mode. In this mode all the agents, including the new one, continue updating its membership vector with the received frames and continue refreshing the RTDB shared areas, too. The T_{xwin} value, however, is not yet adapted and thus the new agent has no slot in the round. When all the team members reach the same membership vector, the number of active team members is updated, so as the inter-slot period T_{xwin}. The protocol enters then in the scan mode in which the agents, using their slightly different values of T_{tup}, rotate their relative phases in the round until they find their slots. From then on, all team members are again synchronized. The removal of an absent agent uses a similar process. After a predefined number of rounds without receiving frames from a given agent, each remaining member removes it from the membership vector. The change in the vector leads to a new agreement process similar to described above.

Fig. 6 shows an example of the self-reconfiguration process with the dynamic insertion and removal of agents. It shows the instants at which the packets from the several agents in a team are received in a monitoring station, relative to the start of the round in an arbitrary agent (agent 2 in this case). The line on top shows the reception instants of that agent, which give us an indication of the effective TDMA round duration. Before point A, agent 2 is alone, using a TDMA round with 1 single slot, and at that point agent 4 joins the team and starts transmitting. Agent 2 detects these transmissions and divides the TDMA round in 2 slots, one for each agent. Naturally, the transmissions of agent 4 are outside the respective validity window, thus agent 4 uses a sliding relative phase until it reaches the right slot, at which point it stays synchronized with the team (near flat portions of the graph). At point B, agent 4 left the team, i.e., stopped transmitting. Agent 2 detected this situation and reconfigured the TDMA round to 1 single slot again. The remaining situations are all similar, with agent 4 rejoining at point C and agent 5 at point D, who leave the team at points E and F, respectively. From D to E the TDMA round is configured to 3 slots and after the withdrawal of agent 4, it is reconfigured to 2 slots again. Notice that the mechanisms are fully distributed and all agents execute exactly the same code.

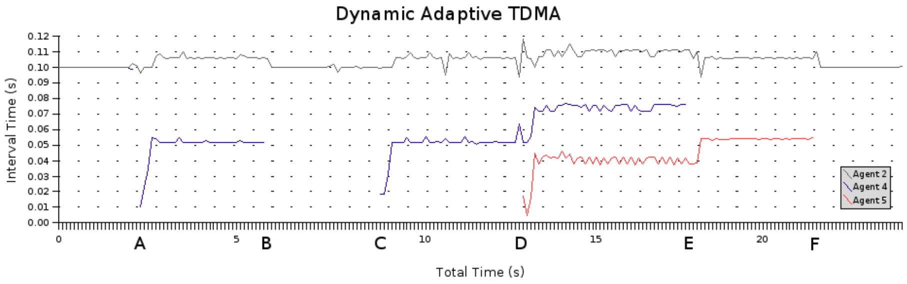


Fig. 6. Self-configuration of the slot time according to the number of running agents

5 Coordinating Multiple Soccer Agents

The purpose of the communication protocol described above is to support the management of the RTDB, which is the central element for sharing information and thus for coordination of the team of agents. In this section we present a brief reference to some of the coordinated behaviors that are currently implemented on top of the RTDB, thus highlighting the effectiveness of the communication.

The team can be in several different play modes, from kick off to free kick, throw in, corner kick, etc., decided by the referee, which are broadcast among the team by the remote station through the RTDB.

The RTDB also supports the integration of the individual agent perceptions to improve their knowledge about the current positions and velocities of the others robots and of the ball. It is very important for our coordination model to keep an accurate estimation of the absolute position of the ball by each robot. The role assignment algorithm is based on the absolute position of the robot, its team mates and ball. Each robot determines its self localization and ball position through its local vision system and shares it with the others through the RTDB.

Communication is also used to convey the coordination status of each agent allowing robots to detect uncoordinated behaviors, for example, several robots with the same exclusive role, and to correct this situation reinforcing the reliability of coordination algorithms.

6 Conclusion

Cooperating robots is a field currently generating large interest in the research community. RoboCup is one example of an initiative developed to foster research in that area.

This paper described the computing and communication architecture of the CAMBADA middle-size robotic soccer team being developed at the University of Aveiro. Such architecture is based on a partially replicated real-time database, i.e., the RTDB, which includes local state variables together with images of remote ones. These images are updated transparently to the application software by means of an adequate real-time management system. Moreover, the RTDB is accessible to the application using a set of non-blocking primitives, thus yielding a fast data access.

Earlier work from the authors led to the development of a wireless communication protocol that reduces the probability of collisions among the team members. The protocol called adaptive TDMA, adapts to the current channel conditions, particularly accommodating periodic interference patterns. In this paper the authors extended that protocol with on-line self-configuration capabilities that allow reconfiguring the slots structure of the TDMA round to the actual number of active team members, further reducing the collision probability. This paper ends with a brief reference to global team coordination based on the RTDB concept, using the described communication protocol.

Future work includes the further dynamic reconfiguration of the TDMA round interval, i.e., the team update period, according to the communication channel load and current number of agents in the team.

References

1. Pedreiras, P., Almeida, L.: Task management for soft real-time applications based on general purpose operating systems. In: Proceedings of the 9th Brazilian Workshop on Real-Time Systems, Belém, Brazil (May 2007)

2. Gopalan, K.: Real-time support in general purpose operating systems. Technical report (2001)
3. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
4. Dietl, M., Gutmann, J.S., Nebel, B.: Cooperative sensing in dynamic environment. In: Proceedings of the IROS2001 International Conference on Intelligent Robots and Systems, Maui, Hawaii (October 2001)
5. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The robot world cup initiative. In: Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife, Montreal (August 1995)
6. Weigel, T., Gutmann, J.S., Nebel, B., Muller, K., Dietl, M.: Cs freiburg: Sophisticated skills and effective cooperation. In: Proceedings of the EEC01 European Control Conference, Porto, Portugal (September 2001)
7. Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, R.: The hersay-ii speech understanding system: Integrating knowledge to resolve uncertainty. ACM Computing Surveys **12**(2) (1980) 213–253
8. Carver, N., Lesser, V.: The evolution of blackboard control architectures. Technical Report UM-CS-1992-071 (1992)
9. Milutinovic, V., Stenstrom, P.: Special issue on distributed shared memory systems. In: Proceedings of the IEEE. Volume 87. (March 1999) 399–404
10. Kopetz, H.: Real-Time Systems Design Principles for Distributed Embedded Applications. Kluwer Academic Pub (1997)
11. Decotignie, J.D., Dallemagne, P., El-Hoiydi, A.: Architecture for the interconnections of wireless and wireline fieldbus. In: Proceedings of the FeT01 IFAC Conference on Fieldbus Technologies, Nancy, France (November 2001)
12. Santos, F., Almeida, L., Pedreiras, P., Lopes, L.S., Facchinetti, T.: An adaptive tdma protocol for soft real-time wireless communication among mobile autonomous agents. In: Proceedings of the WACERTS04 Workshop on Architectures for Cooperative Embedded Real-Time Systems (in conjunction with RTSS2004 - 3rd International Symposium on Robotics and Automation), Lisbon, Portugal (December 2004)

Apêndice E

CAMBADA’2007: Team Description Paper

CAMBADA’2007: Team Description Paper

J. L. Azevedo, N. Lau, G. Corrente, A. Neves, M. B. Cunha, F. Santos,
A. Pereira, L. Almeida, L. S. Lopes, P. Pedreiras, J. Vieira,
P. Fonseca, D. Martins, N. Figueiredo, J. Puga, J. Taborda

Transverse Activity on Intelligent Robotics
IEETA/DETI – Universidade de Aveiro
3810-193 Aveiro, Portugal

Abstract. The CAMBADA middle-size robotic soccer team is described in this paper for the purpose of qualification to RoboCup’2007. The robots have been developed from scratch in the last four years and, unlike other approaches, using home-made mechanical parts and basic electronic modules. Previous experience of some elements of the team in the RoboCup Simulation League has been highly relevant particularly in the design of the high-level coordination and control framework.

1 Introduction

CAMBADA[1] is the RoboCup middle-size league soccer team of the University of Aveiro, Portugal. This project started officially in October 2003 and, since then, the team has participated in three RoboCup competitions, namely, RoboCup’2004, RoboCup’2006, DutchOpen’ 2006, and in the last three editions of the Portuguese Robotics Festival (Robotica2004, Robotica2005 and Robotica2006).

This paper describes the current development stage of the team and is organized as follows: Section 2 briefly presents the robot platform. Section 3 describes the general architecture of the robots focusing both on low-level control hardware aspects and on the general software architecture. Section 4 presents the current version of the vision system. Section 5 briefly describes the high-level coordination and control framework. Finally, section 6 concludes the paper.

2 Robot Platform

The CAMBADA robots were designed and completely built in-house. Each robot is built upon a circular aluminum chassis (with roughly 485 mm diameter), which supports three independent motors (allowing for omnidirectional motion), an electromagnetic kicking device and three NiMH batteries. The remaining parts of the robot are placed in three higher layers, namely: the first layer upon the chassis is used to place all the electronic modules such as motor controllers; the second layer contains the PC (currently a 12" notebook based on an Intel Core2Duo processor); finally on the top of the robots stands an omnidirectional vision system consisting of a standard low cost camera and an hyperbolic mirror (AIS Fraunhofer-Gesellschaft).

---

[1] CAMBADA is an acronym of *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*.

The mechanical structure of the robot is highly modular and was designed to facilitate maintenance. It is mainly composed of two tiers: 1) the *mechanical* section that includes the major mechanical parts attached to the aluminum plate (e.g. motors, kicker, batteries); 2) the *electronic* section that includes control modules, the PC and the vision system. These two sections can be easily separated from each other, allowing an easy access both to the mechanical components and to the electronic modules.

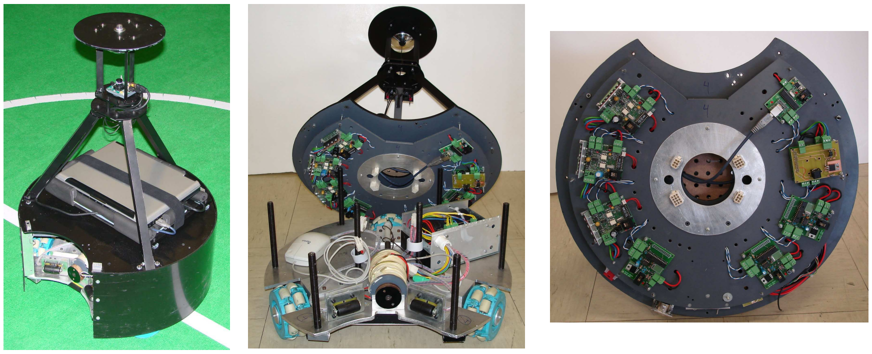


Fig. 1. The CAMBADA robot

3 General Architecture of the Robots

The general architecture of the CAMBADA robots has been described in [1], [2]. Basically, the robots architecture is centered on a main processing unit that is responsible for the higher-level behavior coordination, i.e. the coordination layer. This main processing unit (a PC) processes visual information gathered from the vision system, executes high-level control functions and handles external communication with the other robots. This unit also receives sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system. The PC runs the Linux operating system over the RTAI (Real-Time Applications Interface [6]) kernel, which provides time-related services, namely periodic activation of processes, time-stamping and temporal synchronization. The communication among team robots uses an adaptive TDMA transmission control protocol [3], on top of IEEE 802.11b, that reduces the probability of transmission collisions between team mates thus reducing the communication latency.

The low-level sensing/actuation system (Fig. 2) is implemented through a set of microcontrollers interconnected by means of a network. For this purpose, Controller Area Network (CAN) [5], a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [4],[8]. The low-level sensing/actuation system executes four main functions, namely, Motion control, Odometry, Kicking and System monitoring. The Motion control function provides holonomic motion using 3 DC motors. The Odometry function combines the encoder readings from the 3 motors and provides coherent robot displacement information that is then sent to the coordination layer. The Kick function includes the control of an electromagnetic kicker and of a ball handler to dribble the ball. Finally, the System monitor function monitors the robot batteries as well as the state of all nodes in the low-level layer.

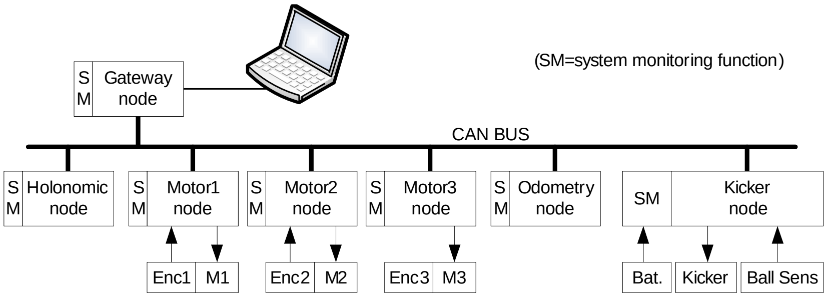


Fig. 2. The CAMBADA hardware architecture.

The low-level control layer connects to the coordination layer through a gateway, which filters interactions within both layers, passing through the information that is relevant across the layers, only.

3.1 Hardware

The low-level layer has a set of nodes, built around a common module, using specialized interfacing to the robot I/O devices. These nodes are interconnected with a CAN network operating at a bit rate of 250Kbps. A gateway interconnects the CAN network to the PC at the high-level layer either through a serial port or a USB port, operating at 115Kbaud in any case.

All modules are based on the same underlying hardware, e.g. a PIC18Fxx8 Microchip [7] microcontroller (@40MHz, i.e., 10 MIPS) which, along with a set of useful peripherals, such as timers, PWM generators, analog to digital converter and serial communications, also integrates a CAN controller. The basic structure of every module includes the CAN port to connect to the network and also includes a 115 Kbps RS232 serial port, which is useful both to program the module firmware and for debugging purposes.

One important characteristic of the CAMBADA hardware design is the galvanic decoupling between the *logic* blocks and the *power* blocks carried out through opto-couplers and/or isolation amplifiers. Along with improved reliability of the whole system it prevents serious damages in expensive equipment (such as the notebook in the high-level layer) whenever any electric problem occurs in the *power* block. The drawback of this solution is the need of an extra battery for the *logic* part of the system.

The main functions implemented in the low-level layer are described in the following.

Motion control

The robot holonomic motion is obtained combining the speed of 3 DC motors (24V-150W), each with its own speed controller. Each of these controllers is a distinct module of the whole distributed architecture implementing a PI closed loop speed control. It takes as inputs the motor shaft displacement, obtained through a quadrature incremental optical shaft encoder coupled to the motor, and the speed set-point. The computation of the three set-points needed to obtain a coherent robot motion is carried out by another module called *holonomic*. It receives the robot velocity vector (speed, direction and heading) from the higher-level (through the gateway) and translates it into individual set-points that are then sent to each motor controller via CAN messages.

Odometry

The odometry function of the robot is accomplished through the combination of 4 basic functions: the reading of the 3 encoders plus their combination to generate coherent displacement information (Δx, Δy, $\Delta\theta$). The reading of each encoder is naturally allocated to each motor module, using the same readings as those used by the speed feedback control. The combination of the readings is carried out in a specific module, the *odometry node*, which receives the encoder readings from the motors and sends the results to the gateway via CAN messages.

Kicking control

The kicking system is based on an electromagnetic kicker that has been developed by the team for these robots. It allows the higher-level coordination functions to choose one of two kicking modes: direct shooting or lofted kick. Effective control of the kick power is also implemented. The kicking system also includes two IR sensors implemented as an IR barrier which is used to detect the ball when it is in the kicking position, thus avoiding false triggering; and a short distance IR sensor (less than 50 cm) which can be used, in addition to visual information, to determine more precisely the distance between the front of the robot and the ball.

Another feature implemented in this module is an active ball-handler system whose purpose is to dribble the ball throughout the game field in accordance with the RoboCup MSL rules. It is implemented as a quadrature incremental encoder, to measure the ball movement thus providing ball rotation feedback control.

System monitoring

This functionality has two main purposes: measure batteries voltage and monitor modules run-time status. The latter requires this function to be present in all modules, tracking reset situations, namely power-up reset, warm reset, brown-out reset (caused by undervoltage spikes) and watchdog reset, as well as answering to *I'm alive* requests issued by the high-level layer. Battery voltage monitoring is implemented in the same module as the kicker, since it already includes specific voltage monitoring hardware. The battery monitoring function measures, in real-time, the voltage of the three NiMH batteries used in the robot, namely 2x12V for the *power* blocks of motor controllers and kicker, plus a 9.6V for the *logic* blocks.

The information gathered by the system monitoring function, in all nodes, is sent to the high-level layer for remote monitoring purposes.

3.2 Software

The software system in each robot is distributed among the various computational units. High level functions run on the PC, while low level functions run on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1], [3], [9] has been adopted. The RTDB is a data structure where the robots share their world models. It is updated and replicated in all players in real-time.

The high-level processing loop starts by integrating perception information gathered locally by the robot. This includes information coming from the vision system and odometry information coming from the low-level layer, both stored in a Local Area of the RTDB. After integration, the world state can be updated in the shared area of the RTDB. The next step is to integrate local information with information shared by team-mates, which is updated by a process that handles the communication with the other robots via an IEEE 802.11b wireless connection. The RTDB is then used by another set of processes that define the specific robot behavior for each instant, generating commands that are passed down to the low-level control layer.

4 Vision System

The current version of the vision system is based on a catadioptric configuration implemented with a low cost Fire-wire web-camera (BCL 1.2 Unibrain camera with a ¼" CCD sensor and a 3.6mm focal distance lens) and a hyperbolic mirror. The camera delivers 640x480 YUV images at a rate of 30 frames per second.

The vision software has been implemented following a modular multi-process architecture (Fig. 3).

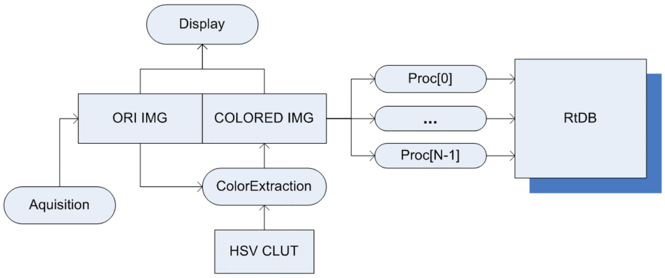


Fig. 3. Architecture of the vision system.

When a new frame is ready to be read, the acquisition process is automatically triggered and the frame is placed in a shared memory buffer. Another process will then analyze the acquired image for color classification, creating a new one with "color labels" (an 8 bit per pixel image). This image is also placed in the shared image buffer, which is afterwards analyzed by the object detection processes, generically designated by *Proc[x]*, x={0, 1,... N-1}. The output of the detection processes is placed in the real-time database (RTDB) which can be accessed by any other processes on the system, such as the world state update. The activation of the different image-processing processes is carried out by means of a process manager [9].

Image analysis in the RoboCup domain is simplified, since objects are color coded. This fact is exploited by defining color classes, using a look-up-table (LUT) for fast color classification. The table consists of 16777216 entries (24 bits: 8 bits for red, 8 bits for green and 8 bits for blue), each 8 bits wide, occupying 16 MB in total. The classification of a pixel is carried out using its color as an index into the table. The color calibration is done in HSV (Hue, Saturation and Value) color space. In the current setup the image is acquired in RGB or YUV format and is then converted to HSV using an appropriate conversion routine.

The image processing software uses radial search lines to analyze the color information. The regions of the image that have to be excluded from analysis (such as the robot itself, the sticks that hold the mirror and the areas outside the mirror) are ignored through the use of a previously generated image mask.

The objects of interest (a ball, two goals, obstacles and the green to white transitions) are efficiently detected through algorithms that, using the color information collected by the radial search lines, calculate the object position and/or their limits in an angular representation (distance and angle). The green/white detected transition points, that are at a distance smaller than a predefined value, are stored in the RTDB for latter use by the robot self-localization process.

The relationship between image pixels and real world distances is obtained through an analytical method developed by the team (to be published soon) that explores a back-propagation ray-tracing approach and the mathematical properties of the mirror surface.

5 High-level coordination and control

The high-level decision is built around three main modules: sensor fusion, basic behaviors and high-level decision and cooperation. The objective of the sensor fusion module is to gather the noisy information from the sensors and from other robots and update the World State database that will be used by the high-level decision and coordination. The basic behaviors module provides the set of primitives that the higher-level decision modules use to control the robot. It is essential to provide those modules with a good set of alternatives, each of which should be as efficient as possible. The high-level decision module is responsible for the analysis of the current situation and for the performing of decision-making processes carried out by each player in order to maximize, not only the performance of its actions, but also the global success of the team.

The sensor fusion module has recently been redesigned, in what concerns its interface with the other modules, in order to get a common view over all the sensor measures. Now all sensors write into adequate structures, but only the sensor fusion module is allowed to update the World State. A recent, and very important, development as been the integration into the sensor fusion module of a self-localization lines-based engine, based-on the one described in [12], that allows a high level of confidence in the robots estimated self-position.

The new design of the vision system, which is now omnidirectional, has allowed the development of a new set of basic behaviors. The previous vision system was based on two cameras, one facing the field orthogonally, enabling the capture of a 360 degrees view around the robot with roughly 1m radius, and the other pointing forward in the direction of the front of the robot. With that vision system the robot could sense far objects in front of it, but had a very limited view of its surrounding area in all other directions. As a consequence most of the movements had to be done with the robot turned to the target point. Using the new vision system, the robot can accurately move towards any given point at any given orientation. Several experiments of different alternatives have been carried out and a new set of optimized basic behaviors is now available.

The high-level decision module currently uses state-machine based modeled roles that switch the basic behavior of the robot in accordance with the current situation and the previous state. Coordination is achieved by the definition of formations of different roles [11] and by a higher-level module where role switching is performed. The concepts of roles, formations and set-plays have previously been used in the RoboCup in some Simulation and Middle-Size teams. The coordination is in the process of integrating the information coming from the new self-localization engine, which allows the use of coordination techniques like SBSP [10]. In some cases, such as kick-ins or corners, specific set-plays are activated where a coordinated and synchronized set of basic behaviors is performed by all team robots.

6 Conclusion

This paper described the current development stage of the CAMBADA robots. Since the last submission of qualification material (in January/2006) several major improvements have been carried out, namely: the implementation of a new vision system based on a single camera in a catadioptric configuration; the development of a new tool to calibrate image colors based on the HSV color space; the implementation of vision software processing based on radial sensors; the development of an analytical method to get the relationship between image pixels and real world distances; the implementation and integration of a self-localization algorithm; the re-design of the higher-level coordination and control software; a new kicking device with kick mode selection and power control; the replacement of the lead-acid batteries by smaller and lighter NiMH which allowed for, roughly, 30% robot weight reduction.

References

1. L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, L.S.Lopes, "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project". ISCIS'04, 19th International Symposium on Computer and Information Sciences. 27-29 October 2004, Kemer - Antalya, Turkey.
2. V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, J. Fonseca, "Implementing a distributed sensing and actuation system: The CAMBADA robots case study", IEEE ETFA 2005, Catania, Italy. September 2005.
3. F. Santos, L. Almeida, P. Pedreiras, L.S.Lopes, T. Facchinnetti, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication Among Mobile Computing Agents". WACERTS 2004, Workshop on Architectures for Cooperative Embedded Real-Time Systems (satellite of RTSS 2004). Lisboa, Portugal, 5-8 Dec. 2004.
4. Almeida L., P. Pedreiras, J. A. Fonseca, "The FTT-CAN Protocol: Why and How, IEEE Transactions on Industrial Electronics", 49(6), December 2002.
5. Controller Area Network - CAN2.0, Technical Specification, Robert Bosch, 1992.
6. RTAI for Linux, available at http://www.aero.polimi.it/~rtai/
7. Microchip website, available at www.microchip.com
8. Calha, M. J., J. A. Fonseca, “Approaches to the FTT-based scheduling of tasks and messages”, Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS’04), Vienna, Austria, Sep/2004.
9. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, "Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques", RoboCup Symposium: Papers and Team Description Papers, RoboCup-2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence, Springer, 2006.
10. Reis, L.P. and N. Lau, "FC Portugal Team Description: RoboCup 2000 Simulation League Champion", RoboCup-2000, P. Stone, et al. (edtrs), p. 29-40, Springer Verlag, 2001.
11. Stone, P. and M. Veloso, "Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork", *Artificial Intelligence*, 110 (2), p. 241-273, 1999.
12. Martin Lauer, Sascha Lange and Martin Riedmiller, “Calculating the perfect match: An efficient and accurate approach for robot self-localisation”, In A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, editors, RoboCup 2005: Robot Soccer World Cup IX, LNCS. Springer, 2005

Apêndice F

CAMBADA: Information Sharing and Team Coordination

CAMBADA: Information Sharing and Team Coordination

Nuno Lau, *Member, IEEE*, Luís Seabra Lopes, *Member, IEEE* and Gustavo A. Corrente

***Abstract*— This paper presents the architecture, information sharing and team coordination methodologies of the CAMBADA RoboCup middle-size league (MSL) team. An overview of the software architecture and individual decision capabilities of the agents is also presented. The information sharing and integration strategy is designed to both improve the accuracy of world models and to support the team coordination. Part of the coordination model is based on previous work in the Simulation League, which has been adapted to the MSL environment. With the described design, CAMBADA reached the 1st place in the Portuguese Robotics Open in 2007 and the 5th place in RoboCup 2007 world championship.**

I. INTRODUCTION

ROBOTIC soccer is currently one of the most popular research domains in the area of multi-robot systems. The RoboCup rules and regulations for different robotic soccer modalities are widely accepted and followed. Many robotic soccer projects use RoboCup competitions for testing and validation of the adopted approaches.

In the context of RoboCup, the so-called “middle-size league” (MSL) is one of the most challenging, since robotic players must be completely autonomous and must play in a field of 12 m 18 m [13]. In this modality, teams are composed of at most six wheeled robots with a maximum height of 80 cm and a maximum weight of 40 Kg. The rules of this modality establish several constraints to simplify perception and world modeling. In particular, the ball is orange, the field is green, the field lines are white, the players are black, etc. The duration of a game is 30 minutes, not including a half-time interval of 5 minutes. The referee orders are communicated to the teams using an application called “referee box”. The referee box sends the referee orders to the team through a wired LAN TCP link connected to the base station of each team. It is the team's responsibility to communicate these orders to the robots inside the field via standard wireless LAN. No human interference is allowed during the games except for removing malfunctioning robots and re-entering robots in the game.

Building a team for the MSL is a very challenging task, both at the hardware and software level. To be competitive, robots must be robust and fast and possess a comprehensive set of sensors. At the software level these robots must have an efficient set of low-level behaviors and must coordinate themselves to operate as a team. Coordination in the MSL league is usually achieved through the assignment of different roles to the robots. Typically there is, at least, an attacker, a defender, a supporter and a goalie [21][2]. As the maximum number of robots in each team increases (it is currently 6) and the field becomes larger, more sophisticated coordination techniques must be developed.

In the RoboCup simulation league teams have been using coordination schemes based on a coordination layer that includes Strategy, Tactics and Formations [17][20], coordination graphs [10] and reinforcement learning [19].

CAMBADA is the RoboCup middle-size league soccer team of the University of Aveiro (Fig. 1). This project started officially in October 2003 and was initially funded by the Portuguese research foundation (FCT). Since then, CAMBADA participated in several national and international competitions, including RoboCup world championships, the European "RoboLudens” and the annual Portuguese Open Robotics Festival.

The CAMBADA project aims at fostering the Aveiro university research at several levels of the MSL challenge. Research conducted within this project has led to developments at the hardware level [3], infrastructure level [1][15][16], vision system [14][5], multi-agent monitoring [9] and high-level decision and coordination [4]. This paper is focused on the last of these components.

This paper is organized as follows: Section II presents the hardware and software architectures of CAMBADA players and provides details on the main software components involved in individual decisions of the players, namely roles and behaviors. Section III describes how players share information with teammates and how they integrate shared information. Section IV describes the adopted coordination methodologies. Section V presents the latest results and concludes the paper.

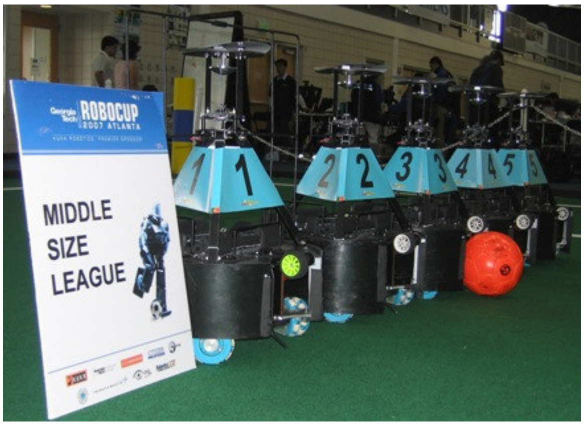


Fig. 1 CAMBADA robotic team

Nuno Lau and Luís Seabra Lopes are with the Transverse Activity on Intelligent Robotics of IEETA research unit as well as with the Department of Electronics, Telecommunications and Informatics, Universidade de Aveiro, Portugal. (e-mails: { nunolau, lsl }@ua.pt).

Gustavo Corrente was a researcher with the Transverse Activity on Intelligent Robotics of IEETA research unit, Universidade de Aveiro, Portugal, and is currently with Nokia Siemens Networks Portugal, Aveiro, Portugal. (e-mail: gustavo@ua.pt)

II. PLAYER ARCHITECTURE

A. *Hardware Architecture*

The CAMBADA robots (Fig. 1) were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter. The mechanical structure of the players is layered and modular. Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromechanical kicker, are attached to an aluminum plate placed 8 cm above the floor. The second layer contains the control electronics. The third layer contains a laptop computer, at 22.5 cm from the floor, and an omni-directional vision system, close to the maximum height of 80cm. The players are capable of holonomic motion, based on three omni-directional roller wheels. The mentioned vision system allows detecting objects (ball, players, goals) and field lines on a radius of nearly 5m around each player. Besides vision, each player includes wheel encoders, battery status sensors and, for detecting if the ball is kickable, an infra-red presence sensor.

The robots computing system architecture follows the fine-grain distributed model [11] where most of the elementary functions, e.g. closed loop control of complex actuators, are encapsulated in small microcontroller based nodes, connected through a network. A laptop node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. USB or Firewire (Fig. 2). For this purpose, Controller Area Network (CAN), a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [7].

In the middle-size league, inter-robot communication and communication between the team's base station and the robots is extremely necessary for the team to maintain a coordinated behavior. The communication among robots and to the base station uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment. The base station is connected to the referee box through a wired LAN TCP link.

B. *Software Architecture*

The software system in each player is distributed among the various computational units (Fig. 2). High-level functions run on the computer, a laptop PC running Linux operating system. Low-level functions run partly on dedicated microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

Fig. 3 shows the class diagram of the CAMBADA WorldState class. This class supports the information storage of ball and players positions, roles, behaviors, etc.. A module called Integrator is used to update the world state information. This is done by filtering the raw information coming from sensors (i.e. vision, odometry, etc.) and determining the best estimate of the position and velocity of each object. The World State class includes several methods that test conditions on the current situation (ex: if the robot is facing the opposite goal).

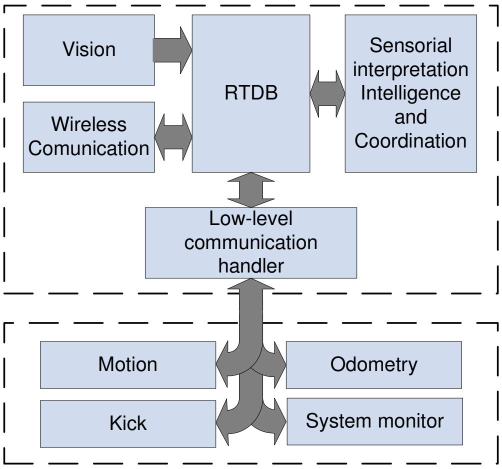

Vision
Wireless Comunication
RTDB
Sensorial interpretation Intelligence and Coordination
Low-level communication handler
Motion
Odometry
Kick
System monitor

Fig. 2. Layered software architecture of CAMBADA players, from [3]

A recent, and very important, development as been the integration into the sensor fusion module of a self-localization lines based engine, based on the one described in [12], that allows a high level of confidence in the robots estimated self position.

The high-level processing loop starts by integrating perception information gathered locally by the player. This includes information coming from the vision processes, which is stored in a Local Area of the RTDB, and odometry information coming from the holonomic base via FTT-CAN. After integration, part of the world state is written in the shared area of the RTDB to make it available to teammates. The next step is to integrate local information with information shared by teammates.

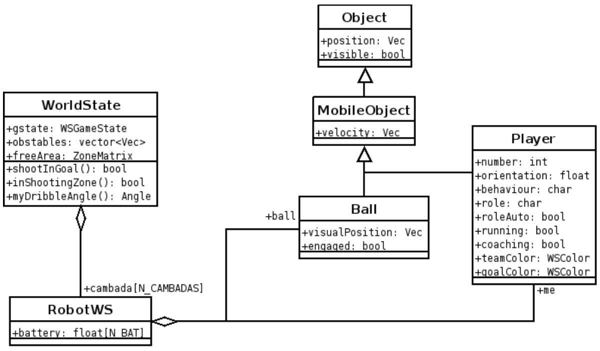

Object
+position: Vec
+visible: bool
MobileObject
+velocity: Vec
WorldState
+gstate: WSGameState
+obstables: vector<Vec>
+freeArea: ZoneMatrix
+shootInGoal(): bool
+inShootingZone(): bool
+myDribbleAngle(): Angle
Player
+number: int
+orientation: float
+behaviour: char
+role: char
+roleAuto: bool
+running: bool
+coaching: bool
+teamColor: WSColor
+goalColor: WSColor
+ball
Ball
+visualPosition: Vec
+engaged: bool
+cambada[N_CAMBADAS]
+me
RobotWS
+battery: float[N_BAT]

Fig. 3. *WorldState* class diagram

The software of the CAMBADA agent is composed of several different processes that have responsibility for different tasks: image acquisition, image analysis, integration/decision and communication with the low-level

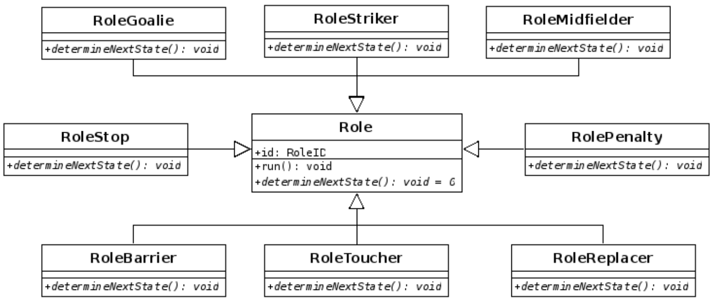

RoleGoalie
+determineNextState(): void
RoleStriker
+determineNextState(): void
RoleMidfielder
+determineNextState(): void
RoleStop
+determineNextState(): void
Role
+id: RoleID
+run(): void
+determineNextState(): void = 0
RolePenalty
+determineNextState(): void
RoleBarrier
+determineNextState(): void
RoleToucher
+determineNextState(): void
RoleReplacer
+determineNextState(): void

Fig. 4. *Role* class diagram

modules. The order and schedule of activation of these processes is performed by a processor manager library called Pman [16]. Pman stores in a database the characteristics of each process to activate and allows the activation of recurrent tasks, settling phase control (through the definition of temporal offsets), precedence restrictions, priorities, etc. The pman services allow changes in the temporal characteristics of the process schedule during run-time.

It is very important that all robots share the same play mode obtained by processing the referee orders given through the referee box. In CAMBADA, an application inside the team's base station checks the messages received from the "referee box", and converts the event triggered protocol of communication "referee box" - "base station" to a state oriented playmode information 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 minimized, enabling a synchronized collective behavior.

C. Roles and Behaviors

The CAMBADA agent decision module is based on the concepts of *role* and *behavior*. Behaviors are the basic sensorimotor skills of the robot, like moving to a specific position or kicking the ball, while roles select the active behavior at each time step.

All roles within a CAMBADA agent are derived from the *Role* abstract class (Fig. 4), whose most important element is the *determineNextState()* method. This method is responsible for the selection of the active behavior. To develop a new role, a *Role* derived class is created and the *determineNextState()* method is implemented. The *run()* method is implemented only in the base class and is responsible for the selection of the active behavior, using *determineNextState()*, and for its execution.

To change the active behavior, the method *changeBehaviour(Behaviour*)*, implemented in the *Role* base class, is used.

During play-on mode, the CAMBADA agents use only three roles: *RoleStriker*, *RoleMidfielder* and *RoleGoalie*. The *RoleGoalie* is activated for the goalkeeper.

RoleStriker is an "active player" role. It tries to catch the ball and score goals according to the finite-state machine shown in Fig. 5. The striker activates several behaviors that try to engage the ball (*MoveToBall*, *MoveOutsideBall*), get into the opponent's side avoiding obstacles (*Dribble*) and shoot to the goal (*Kick*). The *Kick* behavior can perform 180° turns while keeping possession of the ball. The *MoveOutsideBall* is used in situations where a direct catch would lead to the ball getting out of the field. In these situations the robot approaches the ball from the exterior side of the field thus pushing it inside.

RoleMidfielder is a "passive" player. It moves according to its determined strategic positioning [18]. The strategic position is determined for each positioning using a home position and then adjusting it using attractions to the ball current position. Using different home positions and attractions according to the positioning allows a simple definition of defensive, wing, midfielders and attack strategic movement models.

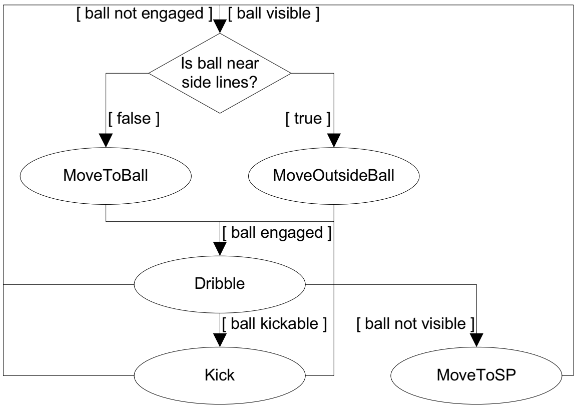

[ball not engaged]
[ball visible]
Is ball near side lines?
[false]
[true]
MoveToBall
MoveOutsideBall
[ball engaged]
Dribble
[ball kickable]
[ball not visible]
Kick
MoveToSP

Fig. 5. Finite-state machine for decision-making in *RoleStriker*

Three more roles are used in set-pieces like kick-off, throw-in, goal-kick, corner-kick, free-kick and penalty. *RoleToucher* and *RoleReplacer* are used to overcome the indirect rule in the case of indirect set pieces. The purpose of *RoleToucher* is to touch the ball and leave it to the *RoleReplacer* player. The replacer handles the ball only after it has been touched by the toucher. This scheme allows the replacer to score a direct goal if the opportunity appears. *RoleBarrier* is used during the set-pieces against CAMBADA to protect the goal from a direct shoot. *RolePenalty* is used in penalty shootouts. It randomly chooses the goal side to which to kick and kicks the ball so that it enters the goal at 0.75m height.

The class diagram of behaviors is shown in Fig. 6. The abstract class *Behavior* is the base of all behaviors. It has three important methods. The first one is *calculate()*, an abstract method, whose implementation in derived classes determines which are the parameters of the command that this behavior intends to execute (*velX*, *velY*, *velA*, *grabberInfo* and *kickerInfo*) but does not execute them. The second is *execute()*, which sends previously computed linear, angular velocities and the kicking parameters to the low level computation modules. The separation of

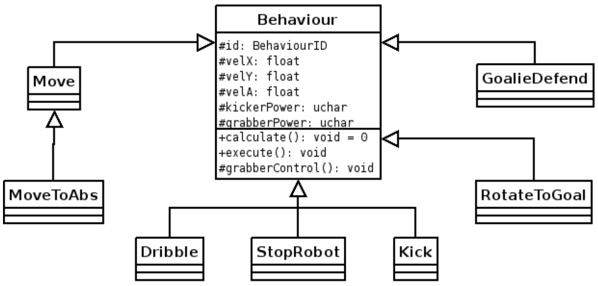


Fig. 6. *Behavior* class diagram

calculation and execution enables the agent to reason on the expected result of the commands while deciding which one to execute. Finally, *grabberControll()* controls the grabber mechanism automatically, without concerns for the behaviors developments.

The set of behaviors that are implemented in the CAMBADA agent are adapted to its catadioptric omnidirectional vision and holonomic driving systems. The combination of these technologies enhances the set of possible behaviors when compared to a differential drive robot or to an holonomic drive robot with a limited field of view.

The behavior *Move* uses two symbolic parameters: the target position where to move; and the position which the CAMBADA player should be facing in its path to the target. The symbols used are *OBall*, *TheirGoal* and *OurGoal*. The other moving behavior *MoveToAbs* allows the movement of the player to an absolute position in the game field. Those moving behaviors may activate the functions of avoiding obstacles and avoiding the ball (used during the game repositions to avoid collisions with the ball). The *Dribble* behavior is used to dribble the ball to a given relative player direction. *GoalieDefend* is the main behavior of the goalie. The *Kick* behavior is used to kick the ball accurately to one 3D position in opponent goal.

III. INFORMATION SHARING AND INTEGRATION

Sharing perceptional information in a team can improve the accuracy of world models. Sharing internal state can improve the team coordination. Therefore, information sharing and integration is one of the key aspects in multi-robot teams.

In CAMBADA, each robot uses some of the perceptions of the other robots, obtained through the RTDB, to improve its knowledge about the current positions and velocities of the others robots and of the ball. It is very important for our coordination model to keep an accurate estimation of the absolute position of the ball by each robot. The role assignment algorithm is based on the absolute positions of the robot and its teammates. The teammates' positions are not obtained through the vision system and rely completely on the communicated estimated self positions of others.

Each agent communicates its own absolute position and velocity to all teammates as well as its ball information (position, velocity, visibility and engagement in robot), current role and current behavior is also shared.

The sharing of own absolute position and velocity is needed first of all because the vision system of the agents currently cannot detect the localization of the teammates. The vision system only detects obstacles but it doesn't try to detect individual robots within the detected obstacles nor does it try to determine if they are teammates or opponents. The absolute position of teammates is necessary to the strategy of our team, as the information is used to define our formation/strategy. So each robot trusts the estimated self position of teammates that is communicated through the RTDB.

Multi-robot ball position integration has been used in the middle-size league by several teams [21][6]. In CAMBADA, multi-robot ball position integration is used to maintain an updated estimate of the ball position, when the vision subsystem cannot detect the ball, and to validate robot's own ball position estimate, when the vision subsystem detects a ball.

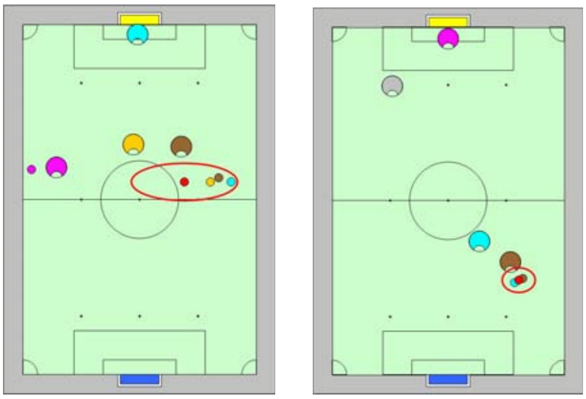

(a) (b)

Fig. 7. Multi-robot ball position integration

Currently, a simple integration algorithm is used. When the agent doesn't see the ball, it analyzes the ball information of playing teammates. The analysis consists in the calculation of the mean and standard deviation of the ball positions, then discarding the values considered as outliers of ball position, and finally using the ball information of the teammate that has a shorter distance to ball. To determine if the agent sees a fake ball, i.e., to validate the robot's own perception, we use a similar algorithm.

Communication is also used to convey the coordination status of each robot allowing robots to detect uncoordinated behavior, for example, several robots with the same exclusive role, and to correct this situation reinforcing the reliability of coordination algorithms.

The communication between the base station and the robots informs the robots of the active playmode (decided by the referee). In some of the used setups, a coach agent, also possibly running in the base station, decides robot's roles and communicates its decisions to the robots using the RTDB. During development the base station can be used to control several robotic agent characteristics like fixed roles, fixed behaviors, manually activated self-positioning, etc, all managed through the RTDB.

IV. TEAM COORDINATION

Our coordination model is based on the definition of a strategy for a game, where each strategy may be composed of several tactics and each tactic defines a formation to be used at each situation in a similar way as SBSP strategies previously developed for the RoboCup Simulation League [18]. However several changes had to be introduced in order to adapt the coordination model to the specificities of the Middle-Size League. This model is merged with role based coordination and different priorities are assigned to the different roles and positionings. In specific situations, like kick ins, or corners, specific set-plays are activated where a coordinated and synchronized set of basic behaviors is performed by all robots in the team.

A. Strategy of Role based strategic positioning

Each tactic is a complete specification of the team coordinated behavior for all situations. A formation defines the movement model of the set of all robots which assigns to each positioning a home position and corresponding attractions to the ball. All these items are maintained in a strategy configuration file, to enable flexible alterations to the current strategy. To maintain a correct formation all robots should have estimations of the ball absolute position that are close to each other. Fig. 8 shows the formation of the team used in Robótica 2007 Tournament [8] for several ball positions.

The *Striker* is helped by other teammates as they maintain their strategic positioning and accompany the striker, without interfering with him, as it plays along the field. In case the ball is captured by the opponent the other mates are in good positions to become the new strikers.

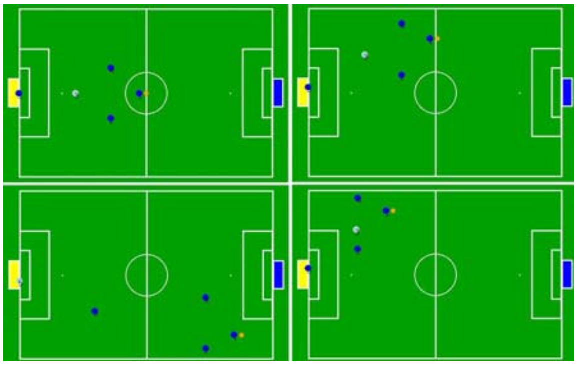


Fig. 8. Strategic positions for several different ball position

B. Role/Positioning assignment algorithm

So far, several different roles have been described but coordination must ensure a proper and safe role assignment algorithm. This algorithm should be able to function with a varying number of active players in the team, either because of hardware or software malfunctioning or because of referee orders. These are very common situations in the MSL.

The playon decision that assigns the *Striker* role and the positionings of the other robots in the formation is performed using an algorithm similar to DPRE [17], but with the innovation of considering different priorities for the different roles and positionings, so that the most important ones are always covered as the number of active players varies.

The algorithm is presented in Fig. 9. Considering a team of *R* field players (not counting the goal-keeper which has a different mechanical configuration and therefore a fixed role), to assign the role and positioning to each robot, the distances of all robots to all strategic positions are calculated. Then the Striker role is assigned to the robot that is closest to the highest priority strategic position, which is in turn the closest to the ball. From the remaining *R-1* robots the closest to the defensive positioning (second highest priority) is assigned to this positioning, then the closest to the third level priority positioning is assigned next and the algorithm continues until all active robots have positionings and roles assigned. This algorithm results in the Striker role having top priority, followed by the defensive positioning, followed by the other supporter positionings. The assignment algorithm may be performed by the coach agent in the base station, assuring a coordinated assignment result, or locally by each robot, in which case the inconsistencies of world models may lead to unsynchronized assignments.

```
MSL_DPRE(robotPositions, ballPosition,
         formation)

clear assignments
determine strategicPositions[N_POSITIONS]
determine distSP[N_POSITIONS][N_ROBOTS]
for each SPos sorted by priority
   determine closest free Agent to SPos
   assign SPos to Agent

Return assignments
```

Fig. 9. CAMBADA Positioning/Role assignment algorithm

C. Set plays

One other coordination methodology that is being used in CAMBADA is the use of predefined set plays. Currently set plays are only initiated when re-entering the play-on mode. Set plays define a sequence of behaviors for several robots in a coordinated way. Each of the tasks that compose a set play are implemented using a special role. These roles are activated at the specific situation: kick-off, kick in, corners, free kicks and goal kicks.

The assignment of the *Barrier*, *Replacer* and *Toucher* roles is executed by sorting the agents according to their perceived distances to the ball and selecting the closest ones, up to the maximum number of agents in each role. When selecting a role like the *Replacer*, which is exclusive, the agent looks at the other teammates role decisions and if it finds a *Replacer* with a lower uniform number it will never select the *Replacer* role. A similar approach is performed for the other roles. This assignment is always performed locally by each robot.

V. CONCLUSION

The data structures used for world state representation clearly separate the raw sensor information from the world model that results of integrating local and shared information. This architecture is easily adaptable to the addition of new sensors. Access to the world model is performed by using specific queries.

The adaptation of SBSP and DPRE [17][18] to the Middle-Size League environment resulted in a coordinated behavior of the team that contributed to its recent successes. The formation flexibility and adaptability was one of the components presented by CAMBADA in the 2nd Technical Challenge of RoboCup 2007 (based on Challenge 6 of [13]), Atlanta, where the team ranked in the 4th place. The robot malfunctions decrease the number of field players, but the positioning/role assignment algorithm maintains a competitive formation with fewer players in the field. Set plays were very efficient as several of the CAMBADA goals were the direct result of their activation.

The work described in this paper was used in two RoboCup competitions:

a. Portuguese Robotics Open 2007 (Portugal): 1st place, 6 wins, 0 draws, 0 looses, 16 goals scored and 3 goals suffered;

b. RoboCup2007 (USA): 5th place, 7 wins, 1 draws, 1 looses, 24 goals scored and 7 goals suffered.

ACKNOWLEDGMENT

The authors and the other CAMBADA team members thank Stefen Welker (Tribots team member) for the supplied material on the localization module, it was very useful to improve our team. Part of this work was funded through a research scholarship granted by IEETA.

REFERENCES

[1] Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes, Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences -- ISCIS 2004: 19th International Symposium, Proceedings*, Aykanat, Cevdet; Dayar, Tugrul; Korpeoglu, Ibrahim, eds., Lecture Notes in Computer Science, Vol. 3280, 2004, pp. 876-886.

[2] M. Arbatzat, S. Freitag, M. Fricke, R. Hafner, C. Heermann, K. Hegelich, A. Krause, J. Krüger, M. Lauer, M. Lewandowski, A. Merke, H. Müller, M. Riedmiller, J. Schanko, M. Schulte-Hobein, M. Theile, S. Welker, D. Withopf: Creating a Robot Soccer Team from Scratch: the Brainstormers Tribots, *Proceedings of Robocup 2003, Padua*, Italy, 2003.

[3] Azevedo, J.L., M.B. Cunha, L. Almeida, Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. Proc. *ETFA2007- 12th IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, 2007, pp. 973-980.

[4] Bartolomeu, P., L. Seabra Lopes, N. Lau, A. Pinho, L. Almeida, Integração de Informação na Equipa de Futebol Robótico CAMBADA, *Electrónica e Telecomunicações*, 4 (4), Universidade de Aveiro, Portugal, 2005, pp. 467-477.

[5] Cunha, B., J. Azevedo, N. Lau, L. Almeida, Obtaining the Inverse Distance Map from a Non-SVP Hyperbolic Catadioptric Robotic Vision System, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*, LNAI, Springer Verlag, Berlin, 2008.

[6] Ferrein, A., L. Hermanns and G. Lakemeyer, Comparing Sensor Fusion Techniques for Ball Position Estimation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, 2006, pp. 154-165.

[7] Ferreira, J.; Pedreiras, P.; Almeida, L.; Fonseca, J.A. The FTT-CAN protocol for flexibility in safety-critical systems, *IEEE Micro*, 22 (4), 2002, pp. 46-55.

[8] Festival Nacional de Robótica'2007, Paderne, Portugal, http://www.ccvalg.pt/robotica2007/, 2007

[9] Figueiredo, J., Lau, N., Pereira, A. Multi-Agent Debugging and monitoring framework, *Proc. First IFAC Workshop on Multivehicle Systems (MVS'06)*, Brasil, October, 2006.

[10] Kok, J.; Spaan, M. and Vlassis, N., Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50 (2-3), Elsevier Science, 2005, pp. 99-114.

[11] Kopetz, H., *Real-Time Systems Design Principles for Distributed Embedded Applications*, Kluwer, 1997.

[12] Lauer, M., S. Lange and M. Riedmiller, Calculating the perfect match: An efficient and accurate approach for robot self-localisation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., LNCS 4020, Springer, 2006.

[13] MSL Technical Committee 1997-2008, *Middle Size Robot League Rules and Regulations for 2008. Draft Version - 12.2 20071109*, November 9, 2007.

[14] Neves, A.; Corrente, G. and Pinho A., An omnidirectional vision system for soccer robots. *Progress in Artificial Intelligence*, Lecture Notes in Computer Science. Berlin, nº 4874, Springer, 2007, pp. 499-507.

[15] Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup-2005: Robot Soccer World Cup IX*, I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, Berlin, 2006, pp. 371-383.

[16] Pedreiras, P.; Almeida, L., Task Management for Soft Real-Time Applications Based on General Purpose Operating Systems, *Robotic Soccer*, edited by: Pedro Lima, Itech Education and Publishing, Vienna, Austria, 2007, pp. 598-607.

[17] Reis, L.P. and N. Lau, FC Portugal Team Description: RoboCup 2000 Simulation League Champion, *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, et al. eds., LNCS 2019, Springer, 2001, pp. 29-40.

[18] Reis, L.P. and N. Lau, and E.C. Oliveira, Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents, *Balancing Reactivity and Social Deliberation in Multiagent Sytems: From RoboCup to Real Word Applications*, M. Hannenbauer, J. Wendler, and E. Pagello eds., LNAI 2103, Springer-Verlag, 2001, pp. 175-197.

[19] Riedmiller, M. Gabel, T., On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup, *Proceedings of the 3rd IEEE Symposium on Computational Intelligence and Games (CIG 2007)*. IEEE Press, April 2007, pp. 17-23.

[20] Stone, P. and M. Veloso, Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork, *Artificial Intelligence*, vol. 110 (2), 1999, pp. 241-273.

[21] Weigel, T. W. Auerbach, M. Dietl, B. Dümler, J.S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski and M. Thiel, CS Freiburg: Doing the Right Thing in a Group, *RoboCup 2000: Robot Soccer World Cup IV*, P. Stone, G. Kraetzschmar, T. Balch, eds., Springer-Verlag, 2001, pp. 52-63.

Apêndice G

CAMBADA’2008: Team Description Paper

CAMBADA’2008: Team Description Paper

J. L. Azevedo, N. Lau, G. Corrente, A. Neves, M. B. Cunha, F. Santos,
A. Pereira, L. Almeida, L. S. Lopes, P. Pedreiras, J. Vieira,
D. Martins, N. Figueiredo, J. Silva, N. Filipe, I. Pinheiro

Transverse Activity on Intelligent Robotics
IEETA/DETI – Universidade de Aveiro
3810-193 Aveiro, Portugal

Abstract. This paper describes the CAMBADA middle-size robotic soccer team for the purpose of qualification to RoboCup’2008. Last year improvements have been made mostly in the vision system, in the high-level coordination and control and in the information integration and localization. Previous experience of some elements of the team in the RoboCup Simulation League has been highly relevant particularly in the design of the high-level coordination and control framework.

1 Introduction

CAMBADA[1] is the RoboCup middle-size league soccer team of the University of Aveiro, Portugal. This project started officially in October 2003 and, since then, the team has participated in four RoboCup competitions, namely, RoboCup’2004, RoboCup’2006, DutchOpen’ 2006 and RoboCup’2007, and in the last four editions of the Portuguese Robotics Festival: Robotica2004, Robotica2005, Robotica2006 and Robotica2007. CAMBADA middle-size robotic soccer team won the 1st place in the Portuguese Robotics Festival’2007 and ranked 5th in the RoboCup World Championship’2007.

This paper describes the current development stage of the team and is organized as follows: Section 2 describes the general architecture of the robots focusing both on low-level control hardware aspects and on the general software architecture. Section 3 presents the current version of the vision system. Section 4 describes the high-level coordination and control framework and, finally, section 5 concludes the paper.

2 General Architecture of the Robots

The general architecture of the CAMBADA robots has been described in [1], [2], [11]. Basically, the robots architecture is centered on a main processing unit that is responsible for the higher-level behavior coordination, i.e. the coordination layer. This main processing unit (a PC) processes visual information gathered from the vision system, executes high-level control functions and handles the external communication with the other robots. This unit also receives sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system. The PC runs the Linux operating system. The communication among team robots uses an adaptive TDMA transmission control protocol [3],[15] on top of IEEE 802.11b, that reduces the probability of transmission collisions between team mates thus reducing the communication latency. This transmission protocol is used to support a Shared Real Time Database (RTDB), which permits sharing selected state variables between the team mates.

---

[1] CAMBADA is an acronym of *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*.

The low-level sensing/actuation system (Fig. 1) is implemented through a set of microcontrollers interconnected by means of a network. For this purpose, Controller Area Network (CAN) [5], a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with the FTT-CAN (Flexible Time-Triggered communication over CAN) [4],[6] higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance. The low-level sensing/actuation system executes four main functions, namely, Motion control, Odometry, Kicking and System monitoring. The Motion control function provides holonomic motion using 3 DC motors. The Odometry function combines the encoder readings from the 3 motors and provides coherent robot displacement information that is then sent to the coordination layer. The Kick function includes the control of an electromagnetic kicker and of a ball handler to dribble the ball. Finally, the System monitor function monitors the robot batteries as well as the state of all nodes in the low-level layer.

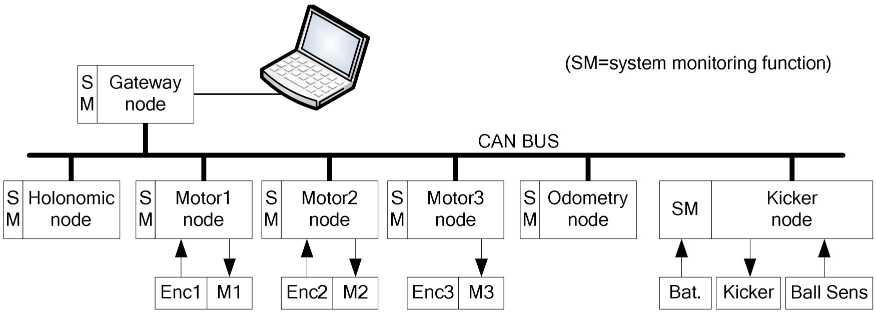


Fig. 1. The CAMBADA hardware architecture.

The low-level control layer connects to the coordination layer through a gateway, which filters interactions within both layers, passing through the information that is relevant across the layers, only.

The software system in each robot is distributed among the various computational units. High level functions are executed on the PC, while low level functions are executed on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1], [3], [7] has been adopted. The RTDB is a data structure where the robots share their world models. It is updated and replicated in all players in real-time.

The high-level processing loop starts by integrating perception information gathered locally by the robot, namely, information coming from the vision system and odometry information coming from the low-level layer. This information is afterwards stored in the local area of the RTDB. The next step is to integrate the robot local information with the information shared by team-mates, disseminated through the RTDB. The RTDB is then used by another set of processes that define the specific robot behavior for each instant, generating commands that are sent down to the low-level control layer.

3 Vision System

Some improvements have been made in the vision system, in particular the development of auto-calibration algorithms and the use of a hybrid vision system integrating an omni-directional and a perspective camera.

The omni-directional part of the vision system [13] is based on a catadioptric configuration implemented with a firewire camera and a hyperbolic mirror. We are using the camera in 640x480 RGB mode at 30 frames per second.

The perspective camera uses a low cost firewire web-camera (BCL 1.2 Unibrain camera with a ¼” CCD sensor and a 3.6mm focal distance lens) configured to deliver 640x480 YUV images at a rate of 30 frames per second.

The omnidirectional vision system is used to find the ball, the goals, detect the presence of obstacles and the white lines (used by the localization algorithm). The perspective vision is used to find the ball and obstacles in front of the robot at higher distances, which are difficult to detect using the omnidirectional vision system.

A set of algorithms have been developed to extract the color information of the acquired images and, in a second phase, extract the information of all objects of interest. To take advantage of the parallel processing capabilities of the hardware, the vision system main tasks, namely, image acquisition, color extraction, object detection and image visualization, are organized in separate processes which, when possible, are executed in parallel (Fig. 2). The implemented color extraction algorithm is based on lookup tables and the object detection in a radial model. The vision system is fast and accurate, having a processing time roughly independent of the environment around the robot.

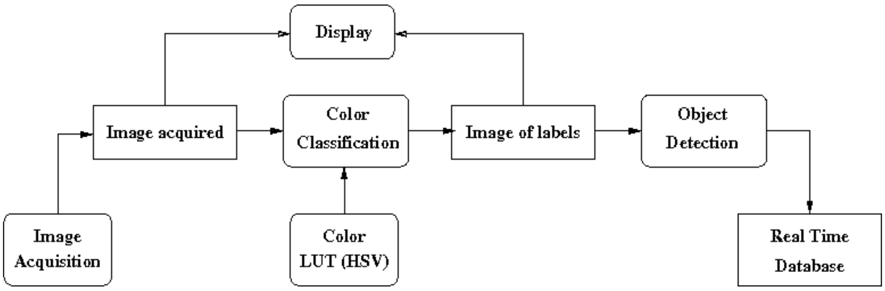


Fig. 2. Architecture of the vision system, applied both to the omnidirectional and perspective subsystem.

Image analysis in the RoboCup domain is simplified, since objects are color coded. This fact is exploited by defining color classes, using a look-up-table (LUT) for fast color classification. The table consists of 16777216 entries (24 bits: 8 bits for red, 8 bits for green and 8 bits for blue), each 8 bits wide, occupying 16 MB in total. The pixel classification is carried out using its color as an index into the table. The color calibration is done in HSV (Hue, Saturation and Value) color space. In the current setup the image is acquired in RGB or YUV format and is then converted to an image of labels using the appropriate LUT.

The image processing software uses radial search lines to analyze the color information. A radial search line is a line that starts in the center of the robot with some angle and ends in the limit of the image. The center of the robot in the omnidirectional subsystem is approximately in the center of the image. However, the center of the robot in the perspective subsystem is in the bottom of the image.

The regions of the image that have to be excluded from analysis (such as the robot itself, the sticks that hold the mirror and the areas outside the mirror) are ignored through the use of a previously generated image mask.

The objects of interest (a ball, obstacles and the white lines) are detected through algorithms that, using the color information collected by the radial search lines, calculate the object position and/or their limits in an angular representation (distance and angle). The position of the ball and the obstacle are stored in the RTDB.

The white lines are detected using an algorithm that, for each search line, finds the transition between green and white pixels. These detected white points are stored in the RTDB for later use by the robot self-localization process.

A set of algorithms have been also developed to perform the auto-calibration of the cameras. These algorithms use a white and a black area to calibrate the values of the white-balance, gain, exposure and brightness. Detailed information about these algorithms will be published soon. The experimental results obtained show the effectiveness of the algorithms, in particular its convergence independently of the original configuration of the cameras and the type of the environment light.

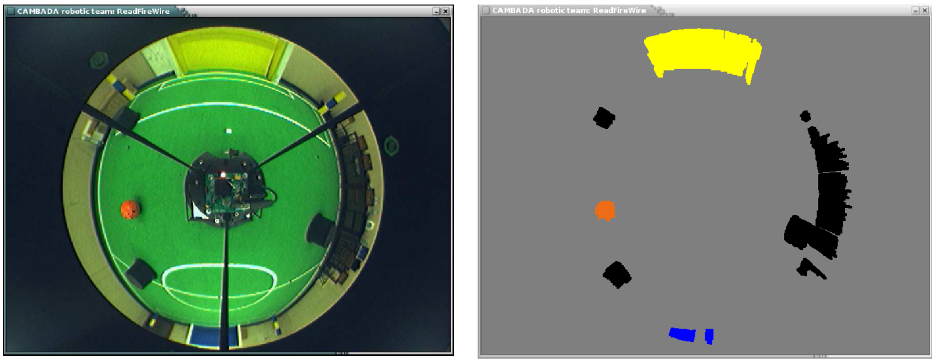

CAMBADA robotic team: ReadFireWire
CAMBADA robotic team: ReadFireWire

Fig. 3. An example of the blobs found in an acquired image. On the left, it is presented the original image. On the right, is it shown the color blobs found for that image. For each blob, we calculate useful information that is used later to calculate the position of each object.

3.1 Inverse Distance Map

The use of a catadioptric omni-directional vision system based on a regular video camera pointed at a hyperbolic mirror is a common solution for the main sensorial element found in a significant number of autonomous mobile robot applications. For most practical applications, this setup requires the translation of the planar field of view, at the camera sensor plane, into real world coordinates at the ground plane, using the robot as the center of this system. In order to simplify this non-linear transformation, most practical solutions adopted in real robots choose to create a mechanical geometric setup that ensures a symmetrical solution for the problem by means of single viewpoint (SVP) approach. This, on the other hand, calls for a precise alignment of the four major points comprising the vision setup: the mirror focus, the mirror apex, the lens focus and the center of the image sensor. Furthermore, it also demands the sensor plane to be both parallel to the ground field and normal to the mirror axis of revolution, and the mirror foci to be coincident with the effective viewpoint and the camera pinhole respectively. Although tempting, this approach requires a precision mechanical setup.

We developed a general solution to calculate the robot centered distances map on non-SVP catadioptric setups, exploring a back-propagation ray-tracing approach and the mathematical properties of the mirror surface [12]. This solution effectively compensates for the misalignments that may result either from a simple mechanical setup or from the use of low cost video cameras. Therefore, precise mechanical alignment and high quality cameras are no longer pre-requisites to obtain useful distance maps. The method can also extract most of the required parameters from the acquired image itself, allowing it to be used for self-calibration purposes.

In order to allow further trimming of these parameters, two simple image feedback tools have been developed.

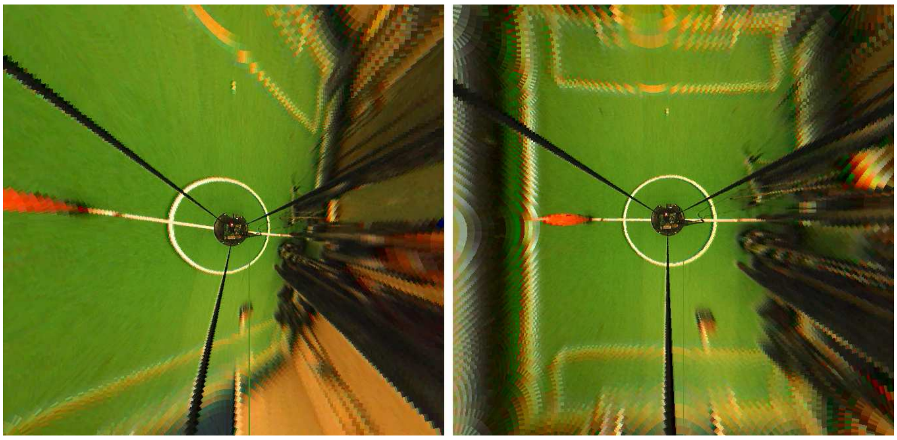


Fig. 4. Acquired image after reverse-mapping into the distance map. On the left, the map was obtained with all misalignment parameters set to zero. On the right, after automatic correction.

The first one creates a reverse mapping of the acquired image into the real world distance map. A fill-in algorithm is used to integrate image data in areas outside pixel

mapping on the ground plane. This produces a plane vision from above, allowing visual check of line parallelism and circular asymmetries (Fig. 4).
The second generates a visual grid with 0.5m distances between both lines and columns, which is superimposed on the original image. This provides an immediate visual clue for the need of possible further distance correction (Fig. 5). Since the mid-field circle used in this setup has exactly an outer diameter of 1m, incorrect distance map generation will be emphasized by grid and circle misalignment.

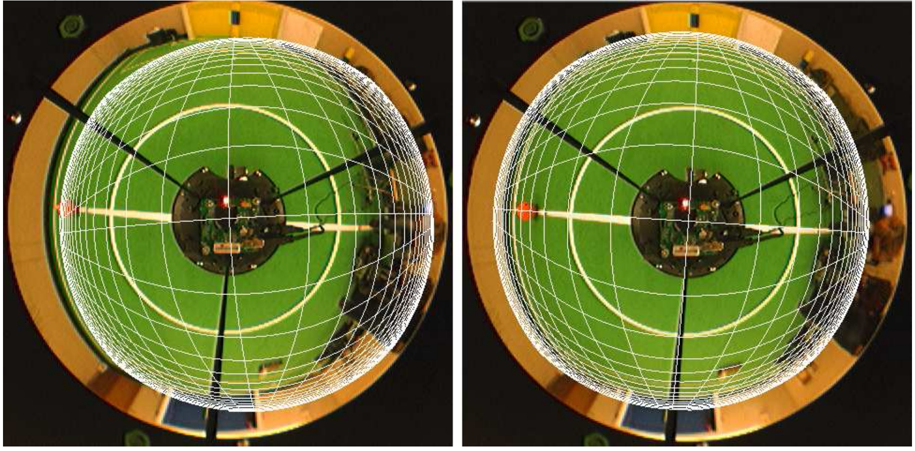


Fig. 5. A 0.5m grid, superimposed on the original image. On the left, with all correction parameters set to zero. On the right, the same grid after geometrical parameter extraction.

4 High-level coordination and control

The high-level decision is built around three main modules: sensor fusion, basic behaviors and high-level decision and cooperation. Monitoring of the whole team of robots is also one of the pursued lines of research [14]. The objective of the sensor fusion module is to gather the noisy information from the sensors and from other robots and update the RTDB database that will be used by the high-level decision and coordination modules. The basic behaviors module provides the set of primitives that the higher-level decision modules use to control the robot. It is essential to provide those modules with a good set of alternatives, each of which should be as efficient as possible. The high-level decision module is responsible for the analysis of the current situation and for the performing of decision-making processes carried out by each player in order to maximize, not only the performance of its actions, but also the global success of the team.

The sensor fusion module has recently been redesigned, in what concerns its interface with the other modules, in order to get a common view over all the sensor measures. Now all sensors write into adequate structures, but only the sensor fusion module is allowed to update the RTDB. A very important development has been the integration into the sensor fusion module of a self-localization lines-based engine, based-on the one described in [10], that allows a high level of confidence in the robots estimated self-position. In order to face the removal of the goal colors, which turned the field into a symmetric environment, an electronic compass has been included to setup the initial orientation of the robots.

The high-level decision module currently uses state-machine based modeled roles that switch the basic behavior of the robot in accordance with the current situation and the previous state. Coordination is achieved by the definition of formations of different roles [9] and by a higher-level module where role switching is performed. The concepts of roles, formations and set-plays have previously been used in the RoboCup by some Simulation and Middle-Size teams. The coordination is in the process of integrating the information coming from the new self-localization engine, which allows the use of coordination techniques like SBSP [8]. In some cases, such as kick-ins or corners, specific set-plays are activated where a coordinated and synchronized set of basic behaviors is performed by all team robots.

4.1 Communication-based Team Coordination

In this environment inter-robot communication and communication between base station and robots is a key issue, so that the team can maintain a coordinated behavior. Each robot uses part of the perception of the other robots, obtained through the RTDB, to improve its knowledge about the current positions and velocities of the other robots and of the ball. It is very important for our coordination model that each robot keeps an accurate estimation of the absolute position of the ball. The role assignment algorithm is based on the absolute position of both the robot and its teammates. The teammates' positions are not obtained through the vision system and rely completely on the communicated estimated self positions of others.

4.2 Multi Robot Ball Position Integration

The CAMBADA team is currently using a simple algorithm for Multi Robot Ball Position Integration. This is used to maintain an updated estimation of the ball position, whenever the vision subsystem is unable to detect the ball, and to validate robot's own ball perception when the vision subsystem detects a ball. When the agent doesn’t see the ball, it analyzes the ball information of playing teammates. The analysis consists in the calculation of the mean and standard deviation of all target ball positions, then discarding the values considered as outliers, and finally using the ball information of the teammate that has a shorter distance to ball.

4.3 Coordination Methodologies

Our coordination model is based on the definition of a strategy for a game, where each strategy may be composed of several tactics and each tactic defines a formation to be used at each situation. This model is merged with a role based coordination where different priorities are assigned to the different roles and positioning. All these items are maintained in a strategy configuration file to enable flexible changes to the current strategy. To maintain a correct formation all robots should have estimations of the ball absolute position obtained through the ball position integration method referred above.

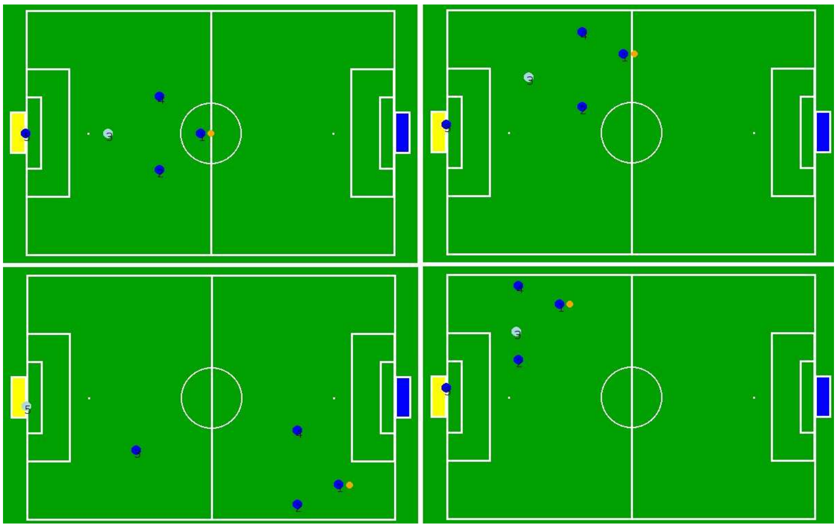


Fig. 6. Strategic positions for several different ball positions.

The role assignment algorithm is designed to support a varying number of active players in the team, resulting either from hardware or software malfunctioning or from referee orders. These are very common situations in the MSL.

5 Conclusion

This paper described the current development stage of the CAMBADA robots. Since the last submission of qualification material (in January/2007) several major improvements have been carried out, namely: the development of auto-calibration algorithms and the use of a hybrid vision system integrating an omni-directional and a perspective camera; the development of an analytical method to get the relationship between image pixels and real world distances and the re-design of the higher-level coordination and control software. These team improvements led to good results both in the Portuguese Robotics Open (1st place) and at RoboCup'2007 Atlanta (5th place). After RopoCup'2007 the development has been focused on perfecting the vision system and the high-level decision algorithms. CAMBADA development team currently includes 5 Msc. students.

References

1. L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, L.S.Lopes, "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project". ISCIS'04, 19th International Symposium on Computer and Information Sciences. 27-29 October 2004, Kemer - Antalya, Turkey.
2. V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, J. Fonseca, "Implementing a distributed sensing and actuation system: The CAMBADA robots case study", IEEE ETFA 2005, Catania, Italy. September 2005.
3. F. Santos, L. Almeida, P. Pedreiras, L.S.Lopes, T. Facchinnetti, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication Among Mobile Computing Agents". WACERTS 2004, Workshop on Architectures for Cooperative Embedded Real-Time Systems (satellite of RTSS 2004). Lisboa, Portugal, 5-8 Dec. 2004.
4. Almeida L., P. Pedreiras, J. A. Fonseca, "The FTT-CAN Protocol: Why and How, IEEE Transactions on Industrial Electronics", 49(6), December 2002.
5. Controller Area Network - CAN2.0, Technical Specification, Robert Bosch, 1992.
6. Calha, M. J., J. A. Fonseca, “Approaches to the FTT-based scheduling of tasks and messages”, Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS’04), Vienna, Austria, Sep/2004.
7. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, "Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques", RoboCup Symposium: Papers and Team Description Papers, RoboCup-2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence, Springer, 2006.
8. Reis, L.P. and N. Lau, "FC Portugal Team Description: RoboCup 2000 Simulation League Champion", RoboCup-2000, P. Stone, et al. (edtrs), p. 29-40, Springer Verlag, 2001.
9. Stone, P. and M. Veloso, "Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork", *Artificial Intelligence*, 110 (2), p. 241-273, 1999.
10. Martin Lauer, Sascha Lange and Martin Riedmiller, “Calculating the perfect match: An efficient and accurate approach for robot self-localisation”, In A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, editors, RoboCup 2005: Robot Soccer World Cup IX, LNCS. Springer, 2005
11. José Luís Azevedo, Manuel Bernardo Cunha, Luís Almeida (2007), Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. ETFA2007- 12th IEEE Conference on Emerging Technologies and Factory Automation, pp. 973 - 980, Patras (Grece) September 25-28, 2007.
12. Bernardo Cunha, José Azevedo, Nuno Lau, Luís Almeida (2007), Obtaining the Inverse Distance Map from a Non-SVP Hyperbolic Catadioptric Robotic Vision System. RoboCup Symposium, Atlanta (USA) July 9-10, 2007 (poster).
13. António J. R. Neves, Gustavo Corrente, Armando Pinho (2007), An omnidirectional vision system for soccer robots. In Progress in Artificial Intelligence, edited by Springer Berlin / Heidelberg, Lecture Notes in Computer Science. Berlin, 2007.
14. João Figueiredo, Nuno Lau, Artur Pereira, Multi-Agent Debugging and monitoring framework, Proc. First IFAC Workshop on Multivehicle Systems (MVS'06), Brasil, October, 2006.
15. Frederico Santos, Gustavo Currente, Luis Almeida, Nuno Lau, Luis Seabra Lopes, Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots, Proc. of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007, Guimarães, Portugal, December 2007.

Apêndice H

The Base Station Application of the CAMBADA Robotic Soccer Team

The Base Station Application of the CAMBADA Robotic Soccer Team

Nuno Figueiredo, António Neves, Nuno Lau, José Azevedo, Artur Pereira and Gustavo Corrente

Abstract – **The base station is the software application responsible to provide automatic processing of soccer game refereeing events and to allow high level monitoring and control of the robots internal states. This paper presents the base station developed for CAMBADA, the robotic soccer team of the University of Aveiro. It describes the main requirements and specifications of the base station and presents the architecture of the application, giving special attention to the description of the main modules and to the connection between them. It also describes the multi-window system, among other issues, namely the classes implemented and the mechanism of passing information among the several modules.**

Resumo – **A estação base é a aplicação de software responsável pelo processamento automático dos eventos que ocorrem num jogo de futebol robótico e pela monitorização e controlo do estado interno dos robôs. O presente artigo descreve a estação base desenvolvida para a equipa CAMBADA, o projecto de futebol robótico da Universidade de Aveiro. Neste artigo são descritos os principais requisitos e especificações da aplicação, bem como a sua arquitectura, dando especial atenção aos principais módulos e à forma como eles se interligam e comunicam entre si. O artigo descreve também o sistema de janelas múltiplas bem como algumas outras questões, nomeadamente, as classes implementadas e o mecanismo de passagem de informação para diversos módulos do sistema.**

I. Introduction

Robotic soccer is nowadays a popular research domain in the area of multi-robot systems. RoboCup[1] is an international joint project to promote research in artificial intelligence, robotics and related fields. RoboCup chose soccer as the main problem aiming at innovations to be applied for socially relevant problems. It includes several competition leagues, each one with a specific emphasis, some only at software level, others at both hardware and software, with single or multiple agents, cooperative and competitive.

In the context of RoboCup, the Middle Size League (MSL) is one of the most challenging. In this league, each team is composed of up to 6 robots with a maximum size of $50cm \times 50cm$ width, $80cm$ height and a maximum weight of 40Kg, playing in a field of $18m \times 12m$. The rules of the game are similar to the official FIFA rules, with minor changes required to adapt them for the playing robots [1].

The rules of this league establish several constraints to simplify perception and world modeling. In particular, the ball is orange, the field is green, the field lines are white and the players are black. The duration of a game is 30 minutes, not including a half-time interval of 5 minutes. The game is refereed by a human and his orders are communicated to the teams using an application called “referee box” operated by an assistant referee. The referee box sends the referee orders to the team through a wired LAN TCP link connected to the external computer of each team. It is the team responsibility to communicate these orders to the robots through the field wireless network.

No human interference is allowed during the games except for removing malfunctioning robots and re-entering robots in the game. Each robot is autonomous and has its own sensorial means. They can communicate among each other and with an external computer through a wireless network. This external computer, that has no sensor of any kind, runs the base station application. The base station only “knows” what is reported by the playing robots and the orders received from the referee box. The agents should be able to evaluate the state of the world and take decisions suitable to fulfill the cooperative team objective.

CAMBADA[2], *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*, is the Middle Size League Robotic Soccer team from the University of Aveiro. The CAMBADA research project started in 2003, coordinated by the Transverse Activity on Intelligent Robotics (ATRI)[3] group of the Institute of Electronic and Telematic Engineering of Aveiro (IEETA)[4]. Since then, it has involved people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers [2] and the software development in areas such as image analysis and processing [3]-[7], sensor and information fusion [8], reasoning and control [9].

Since its creation, the team has participated in several competitions, both national and international. Each year, new challenges are presented, and new objectives are defined, always with a better team performance in sight. After achieving the first place in the national competition Robótica 2007 and Robótica 2008, the 5th place in the world championship RoboCup 2007, this year the team achieved the first place in the world championship RoboCup 2008.

---

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

[2] `http://www.ieeta.pt/atri/cambada`

[3] `http://www.ieeta.pt/atri`

[4] `http://www.ieeta.pt`

This paper presents the base station developed for the robotic soccer team CAMBADA. Being this application of extreme importance for the team, the requirements and specifications of the project had to be carefully analyzed. These issues are presented in Section II. Section III presents the software architecture of the base station. Section IV describes the implementation details, in particular the classes developed and the information update mechanism. Finally, Section V draws some conclusions.

II. Requirements and Specifications

The base station is the software application responsible to provide automatic processing of soccer game refereeing events (coming from the referee box) and to allow high level monitoring and control of the robots internal states. This application must be able to show all relevant information of the robots, namely position in the field, velocity, battery charge, among other information, and send basic commands/information to the robots, in particular the run and stop commands, play mode, etc. Besides that, the base station has a fundamental role in a game, while receiving the commands from the referee box, translating them to internal game states and broadcasting the results to the robots. During a game, no human interference is allowed except for removing malfunctioning robots and re-entering robots in the game.

The role of the base station during the game led to the fulfillment of some requirements, being the most important the following:

Reliability / Stability: during the game, the base station is not accessible for human interaction of any kind and thus, it has to be a very robust application.

Usability: the information displayed in the base station should be easy to interpret, allowing, for instance, for a fast detection of a problem in a robot. Moreover, it should be possible to choose different levels of details in the displayed information.

Usability in the team development stage: the base station has to be easy to use, allowing an intuitive management of the robots.

Adaptability: a robotic soccer team is in a permanent development stage, which may lead to significant changes within a short period of time.

These requirements led to the following specifications:

Modular construction: a robot, for instance, should be an instantiable entity in the application in order, for instance, to allow the inclusion of more robots in the team. This modular construction leads to a progressive development, allowing the test of each module separately, thus increasing the reliability and stability of the whole application.

Multi-windows solution: the application should be a multi-window environment, allowing the user to choose between different levels of information. At least, three different levels of information should be provided: a work level that presents the controls of the robots and basic status information; a visual level that presents visual information of the position of the robots and, finally a detailed level that shows all the information related to the robots.

Robust communication skills: the correct operation of the team during the game is fully dependent on the communication between the robots, the base station and the referee box. Hence, the base station should provide a robust communication layer.

Automatic processing of the game states: the base station should process the commands received from the referee box allowing the robots to change their internal game states accordingly. One specific action should be the changing of the field side at half time.

Adaptable windows geometry: the multi-windows system should adapt to monitors with different resolutions. According to the new (upcoming) MSL rules, the teams base stations must use an external monitor provided by the organizing committee.

In order to use the base station during the team development phase, the following specifications should also be met:

Local referee box: the base station should provide an interface widget to emulate a real referee box in order to simulate events of a real game.

Manual positioning of the robots in the field: it should be possible to move the robots, through a mouse driven operation, to a specific position on the field.

Manual role assignment: acting as a cooperative team, each robot has a specific role which is, during a real game, dynamically assigned. In the development phase, it should be possible to statically assign a role to a specific robot.

III. System Architecture

The central software component in the architecture of the CAMBADA robots is the Real-Time Database (RTDB) [10]. The RTDB is a distributed data structure by means of which all the team agents share their world models. It is updated and replicated in all robots, base station and coach (the coach is a software application, that running in the same computer of the base station, that can, in some specific situations, assign the robots roles). The RTDB contains all information related to the robots and game, namely positions in the field, roles, behaviors, game sates, etc.

All the interaction between base station and the CAMBADA robots is performed through the RTDB. Due to this fact, one of the central modules in the

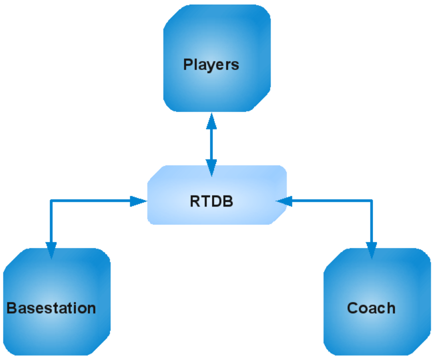


Fig. 1 - All the commands and information exchanged between all the team agents is accomplished through the RTDB.

base station architecture is the one responsible for handling the communication with RTDB, named as `UpdateWidget` (Fig. 6). It creates an abstraction layer to the RTDB interaction mechanism.

Another module in the system is the `RobotWidget` (Fig. 2). This module is responsible to send commands to the robots and shows robots information, such as position in the field, battery charge, etc.

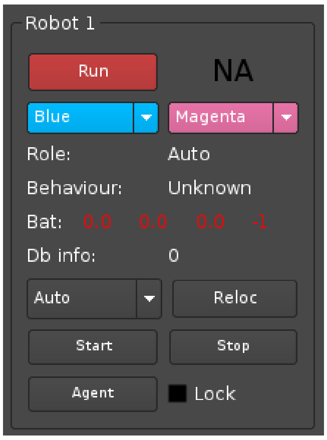

Robot 1
NA
Role: Auto
Behaviour: Unknown
Bat: 0.0 0.0 0.0 -1
Db info: 0

Fig. 2 - The `RobotWidget` module showing the information and the commands available for one robot.

The `FieldWidget` module is responsible to create a visual interface to the user (Fig. 3). This module draws the field and the robots and can provide a simple way to control the position of the robots using a mouse driven interaction.

The `RobotInfoWidget` module shows all the information related to the robots stored in RTDB (Fig. 4).

Another important module in the base station system is the `RefBoxWidget`. This module provides a local referee box for test purposes and also manages the information received from a real referee box during a game. It provides an easy interface to configure the connection to the external referee box and allows the temporary suspension of the handling of the referee box messages maintaining the connection.

The base station implements a three windows based solution, allowing the user to choose the better set of information/actions that he wants to see/perform. When the application starts, it shows one window, the `MainWindow` (Fig. 7). The user can, at any time, open the other windows: the `FieldWindow` (Fig. 8) and the `InfoWindow` (Fig. 4). These two windows can be

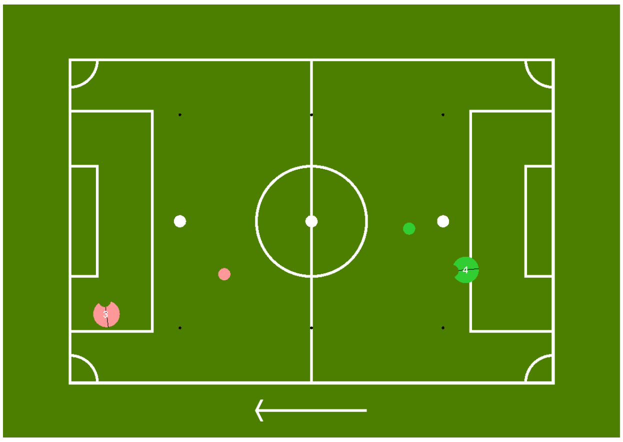
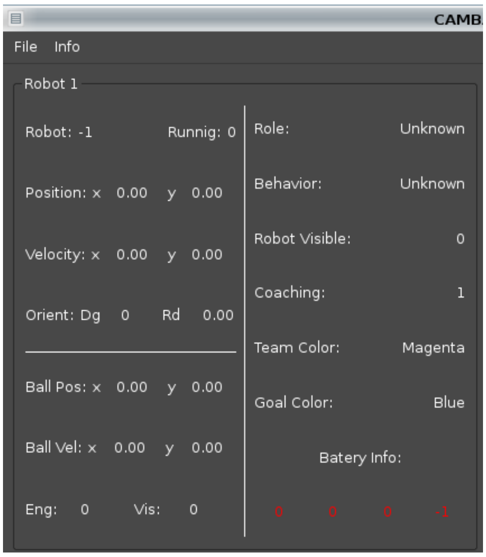
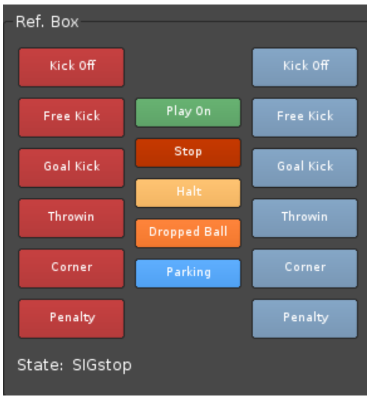


Fig. 3 - The `FieldWidget` showing the position of two robots and the position of the ball as estimated by each of them.



Robot 1
Robot: -1 Runnig: 0
Position: x 0.00 y 0.00
Velocity: x 0.00 y 0.00
Orient: Dg 0 Rd 0.00
Ball Pos: x 0.00 y 0.00
Ball Vel: x 0.00 y 0.00
Eng: 0 Vis: 0
Role: Unknown
Behavior: Unknown
Robot Visible: 0
Coaching: 1
Team Color: Magenta
Goal Color: Blue
Batery Info:
0 0 0 -1

Fig. 4 - The information of one robot in the base station information window. This window shows the information of all robots of the team.


Ref. Box

State: SIGstop

Fig. 5 - The `RefBoxWidget` module providing a local referee box for test purposes.

opened and closed independently. This solution allows the user to have more than one screen with different windows on each one. If the `MainWindow` is closed, all the other windows will also be closed too.

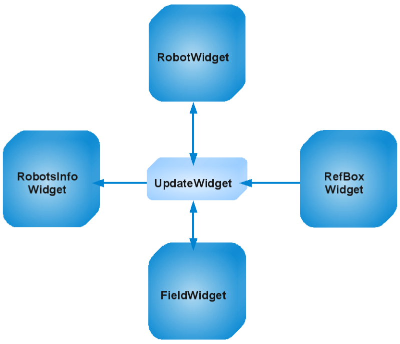


Fig. 6 - The most important base station modules.

IV. IMPLEMENTATION

The base station project was developed in C++ in a Linux environment using the Qt4 libraries [11]. The Qt4 libraries were developed in C++ and provide graphical and communication functions. This section describes the most important classes developed for the base station application and the most relevant issues in the development process.

A. Base station Classes

In the base station project, the most important classes that have been implemented are the following:

`UpdateWidget`: responsible for the connection between the base station and the RTDB. In this class, it is declared a local image of the RTDB that is passed, through a reference based mechanism, to the other classes in the project. This class is responsible for the connection to the RTDB and for the update of the information present in the local data structure.

`FieldWidget`: implemented using the integrated class "GLWidget", offered by the Qt4 library, that merges the Qt4 communication mechanisms and the graphical engine OpenGL. This class is responsible for drawing the field and the robots. This class implements a mechanism that allows the user to select a robot and pass a new point in the field to where the robot should move, using a mouse based mechanism.

`RobotWidget`: this class implements all the visual elements, like buttons, combo boxes, etc. related to the robot information and control. It is also responsible to process the information/commands related to the robots. This class is instantiated as many times as the total number of robots existing in the team.

`RobotInfoWidget`: this class is responsible for the visual elements that show all the information stored in the RTDB. Like `RobotWidget`, this class is instantiated as many times as the total number of robots existing in the team.

`RefBoxWidget`: this class is responsible for the creation, handling and destruction of the connection between the base station and the external referee box during a game. This class is also responsible for processing the game information and to perform the change of the team side at half time. This class also implements a local referee box.

`MainWindowWidget`: this is the application main class. It constructs the other classes, handles the mechanism of communication between UpdateWidget and the other classes and implements all the visual elements concerning team commands (coordinating all the robots at same time). This class manages the other windows.

`FieldWindowWidget`: this class implements the visual elements of the `FieldWindow`.

`InfoWindowWidget`: this class implements the visual elements of the `InfoWindow`.

B. UpdateWidget Mechanism

This mechanism allows sharing the same memory space with all modules in the application. It was implemented using the concept of parent and child, where a parent shares, with its children, the pointer to the memory space.

The `UpdateWidget` includes the method `DB_Robot_Info *get_info_pointer(void)` that returns the pointer to the structure where a local image of RTDB is stored. In all modules that interact with this information, there is a method named `void get_info_pointer( DB_Robot_Info* )` which has to be called to give access to the structure information.

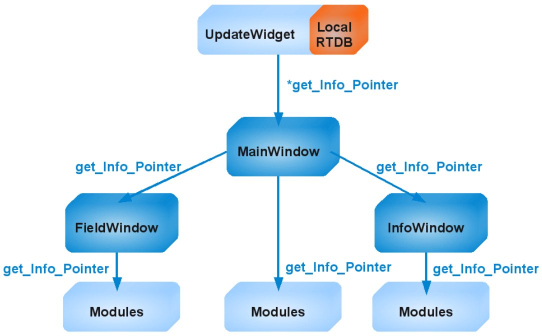
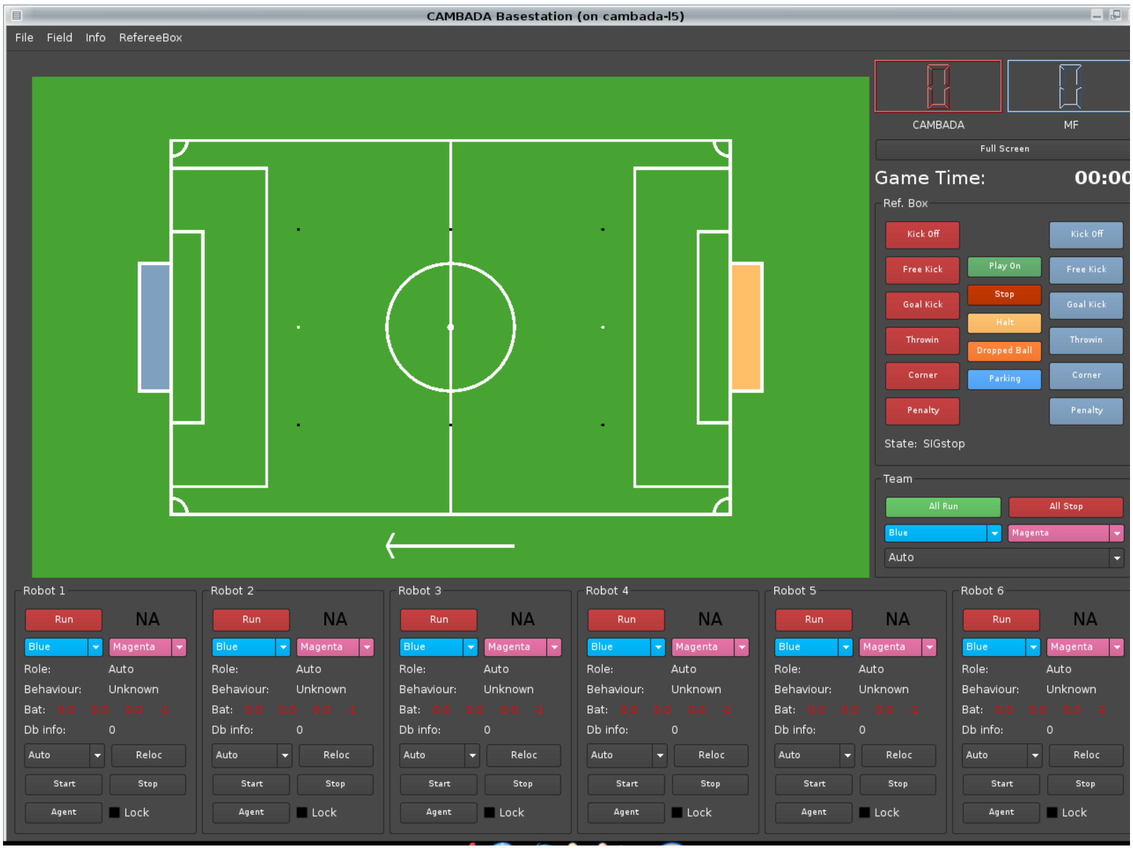


Fig. 9 - The `UpdateWidget` references mechanism.

Figure 9 shows the process of passing the references in all main widgets of the application.

The process begins in the `MainWindow` object. After calling the constructor of the `UpdateWidget`, the method `get_info_pointer` from `UpdateWidget` is called. This function returns a pointer to the memory space. After that, the `MainWindow` class passes this information to all its internal modules and other classes. All the classes pass the pointer to their children.

The implementation of this mechanism rises two main questions:


CAMBADA Basestation (on cambada-l5)
File
Field
Info
RefereeBox
CAMBADA
MF
Full Screen
Ref. Box
Kick Off
Free Kick
Goal Kick
Throwin
Corner
Penalty
Play On
Stop
Halt
Dropped Ball
Parking
Kick Off
Free Kick
Goal Kick
Throwin
Corner
Penalty
Team
All Run
All Stop
Blue
Magenta
Auto
Robot 1
Robot 2
Robot 3
Robot 4
Robot 5
Robot 6
Run
NA
Blue
Magenta
Auto
Reloc
Start
Stop
Agent
Lock

Game Time: 00:00

State: SIGstop

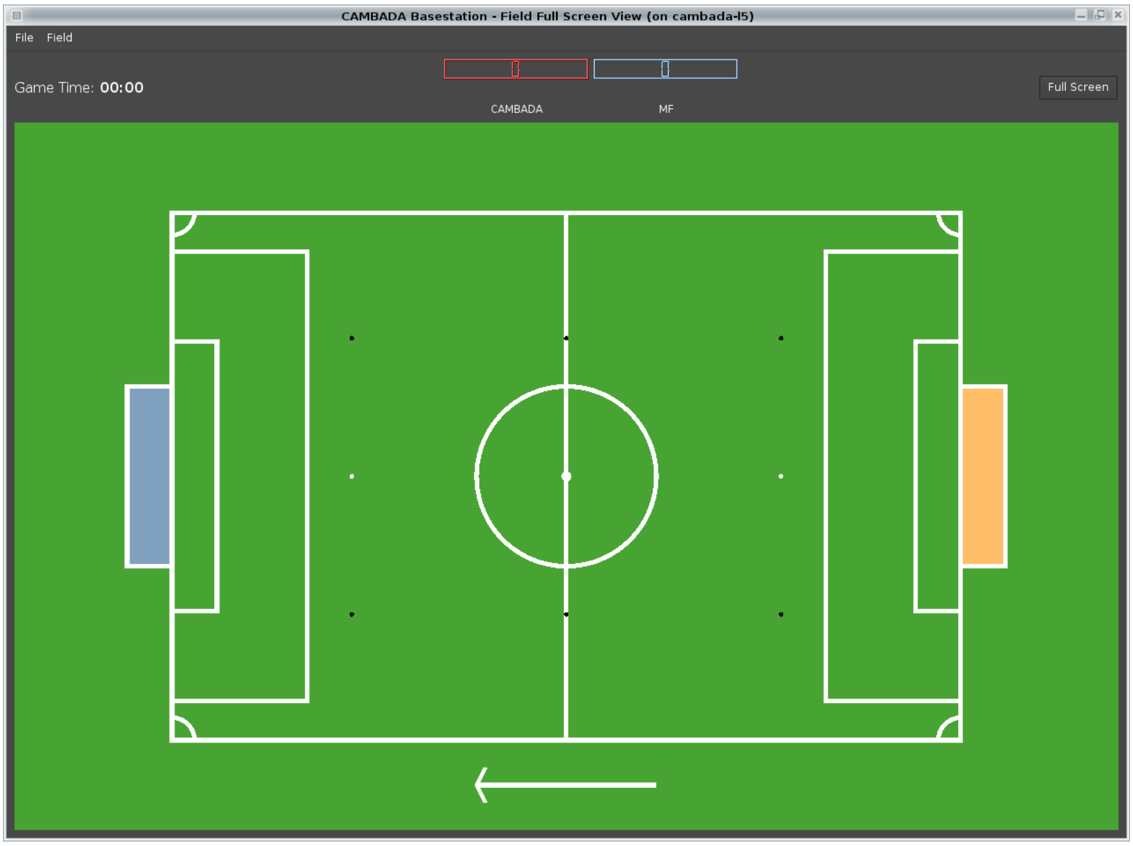
Role: Auto

Behaviour: Unknown

Bat:

Db info: 0

Fig. 7 - The `MainWindow` of the base station application.


CAMBADA Basestation - Field Full Screen View (on cambada-l5)
File
Field
CAMBADA
MF
Full Screen

Game Time: 00:00

Fig. 8 - The `FieldWindow` of the base station application.

Why is this mechanism not included in the constructor of each class?

To include some classes in the design procedure it is mandatory that they have default constructors with predefined input parameters.

Why doesn't this mechanism has problems of mutual exclusion?

All interaction with the memory space is performed inside slot methods (Qt4 mechanism that responds to a specific signal) which guarantees the mutual exclusion (thread-safe) [12].

C. RefBoxWidget incoming messages handling mechanism

One important issue in processing the messages coming from the referee box is to guarantee the consistency of the information in all robots. It is very important that all robots share the same game states (play mode). This is guaranteed by the broadcast mechanism of the RTBD [13], [14].

The internal game states implemented in the CAMBADA are: `Start`, `Stop`, `DropBall`, `OurKickOff`, `TheirKickOff`, `OurPenalty`, `TheirPenalty`, `OurFreeKick`, `TheirFreeKick`, `OurGoalKick`, `TheirGoalKick`, `OurThrowIn`, `TheirThrowIn`, `OurCornerKick` and `TheirCornerKick`.

The base station has the notion of the team color and compares all orders to decide which is the next internal game state. An order like `Magenta Free Kick` could be an `OurFreeKick` if the team has the Magenta color. However, if the team color is cyan, the internal game state will be `TheirFreeKick`.

The referee orders could be classified into two classes: `Game State` orders and `Game Status` orders. The `Game State` orders are concerned with the state of the game. There are orders like `Start, Stop, MagentaGoalKick`, etc. The `Game State` orders have some special requirements to be processed. The order of reception is important and, after each order, the robots have to be informed of each game state. This is more relevant in case of more than one command is received in the same referee box message. However, a status message doesn't have these requirements. If more than one command is sent in the same message, the order of reception will be irrelevant. The algorithm implemented in `RefBoxWidget` reflects that.

V. Conclusions

The application described in this paper was used in the Portuguese Robotics Open "Robótica 2008" where CAMBADA team was, for the second time, national champion. Moreover, it was also used in world championship "RoboCup 2008" where CAMBADA team has won, for the first time, the world champion title. During these events, the application showed a high level of stability and reliability that was identified as a special requirement in the beginning of this project. Besides that, all other requirements were fulfilled, concluding that the base station was an important agent in the success of CAMBADA, contributing to the excellent results obtained by the team in the last year.

Acknowledgment

This work was supported in part by the FCT project PTDC/EIA/70695/2006.

References

[1] MSL Technical Committee 1997-2008, "Middle Size Robot League Rules and Regulations for 2008", 2007.

[2] 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, ETFA2007*, Greece, 2007, pp. 973–980.

[3] A. J. R. Neves, G. Corrente, and A. J. Pinho, "An omnidirectional vision system for soccer robots", in *Proc. of the EPIA 2007*. 2007, vol. 4874 of *Lecture Notes in Artificial Inteligence*, pp. 499–507, Springer.

[4] A. J. R. Neves, D. A. Martins, and A. J. Pinho, "A hybrid vision system for soccer robots using radial search lines", in *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, Aveiro, Portugal, april 2008, pp. 51–55.

[5] D. A. Martins, A. J. R. Neves, and A. J. Pinho, "Real-time generic ball recognition in RoboCup domain", in *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008, IROBOT Workshop*, Lisbon, Portugal, october 2008, pp. 37–48.

[6] P. M. R. Caleiro, A. J. R. Neves, and A. J. Pinho, "Colorspaces and color segmentation for real-time object recognition in robotic applications", *Revista do DETUA*, vol. 4, no. 8, pp. 940–945, June 2007.

[7] B. Cunha, J. L. Azevedo, N. Lau, and L. Almeida, "Obtaining the inverse distance map from a non-svp hyperbolic catadioptric robotic vision system", in *Proc. of the RoboCup 2007*, Atlanta, USA, 2007.

[8] J. Silva, N. Lau, J. Rodrigues, and J. A. Azevedo, "Ball sensor fusion and ball interception behaviours for a robotic soccer team", in *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008, IROBOT Workshop*, Lisbon, Portugal, october 2008, pp. 25–36.

[9] L. S. Lopes N. Lau and G. A. Corrente, "CAMBADA: Information sharing and team coordination", in *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, Aveiro, Portugal, april 2008, pp. 27–32.

[10] L. Almeida, F. Santos, T. Facchinetti, P. Pedreira, V. Silva, and L. S. Lopes, "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project", in *Proc. of the 19th International Symposium on Computer and Information Sciences, ISCIS 2004*. 2004, vol. 3280 of *Lecture Notes in Computer Science*, pp. 878–886, Springer.

[11] Trolltech Co, "Qt cross-platform application framework". **URL:** *http://www.trolltech.com*

[12] Trolltech Co, "Signals and slots across threads". **URL:** *http://doc.trolltech.com/4.0/threads.html#signals-and-slots-across-threads*

[13] F. Santos, L. Almeida, P. Pedreiras, L.S. 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, pp. 657–665.

[14] F. Santos, G. Corrente, L. Almeida, N. Lau, and L.S. 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.

Apêndice I

Coordinated Action in Middle-Size Robotic Soccer

Coordinated Action in Middle-Size Robotic Soccer

Nuno Lau, *Member, IEEE*, Luís Seabra Lopes, *Member, IEEE*, Gustavo Corrente and Nelson Filipe

Abstract — **This paper presents the architecture, information sharing and team coordination methodologies of the CAMBADA RoboCup middle-size league (MSL) team. The information sharing and integration strategy is designed to improve both the accuracy of world models and to support the team coordination. Part of the coordination model is based on previous work in the Soccer Simulation League, which has been adapted to the MSL environment. With the described design, CAMBADA reached the 1st place in the RoboCup 2008 world championship. Competition results and performance measures computed from logs and videos of competition games are presented and discussed.**

I. INTRODUCTION

ROBOTIC soccer is currently one of the most popular research domains in the area of multi-robot systems. The RoboCup rules and regulations for different robotic soccer modalities are widely accepted and followed. Many robotic soccer projects use RoboCup competitions for testing and validation of the adopted approaches.

In the context of RoboCup, the so-called “middle-size league” (MSL) is one of the most challenging, since robotic players must be completely autonomous and must play in a field of 12 m × 18 m [16]. In this modality, teams are composed of at most six wheeled robots with a maximum height of 80 cm and a maximum weight of 40 Kg. The rules of this modality establish several constraints to simplify perception and world modeling. In particular, the ball is orange, the field is green, the field lines are white, the players are black, etc. The duration of a game is 30 minutes, not including a half-time interval of 5 minutes. The referee orders are communicated to the teams using an application called “referee box”. The referee box sends the referee orders to the team through a wired LAN TCP link connected to the base station of each team. It is the team's responsibility to communicate these orders to the robots inside the field via standard wireless LAN. No human interference is allowed during the games except for removing malfunctioning robots and re-entering robots in the game.

Building a team for the MSL is a very challenging task, both at the hardware and software level. To be competitive, robots must be robust and fast and possess a comprehensive set of sensors. At the software level these robots must have an efficient set of low-level behaviors and must coordinate themselves to operate as a team. Coordination in the MSL league is usually achieved through the assignment of different roles to the robots. Typically there is, at least, an attacker, a defender, a supporter and a goalie [27][2]. As the maximum number of robots in each team increases (it is currently 6) and the field becomes larger, more sophisticated coordination techniques must be developed.

In the RoboCup Soccer Simulation League, teams have been using coordination schemes based on a coordination layer that includes strategy, tactics and formations [22][25], coordination graphs [11] and reinforcement learning [23].

CAMBADA is the RoboCup middle-size league soccer team of the University of Aveiro (Fig. 1). The project aims at fostering the Aveiro university research at several levels of the MSL challenge. Research conducted within this project has led to developments at the hardware level [3], infrastructure level [1][19][20], vision system [17][4], multi-agent monitoring [9] and high-level decision and coordination [13]. This paper is focused on the last of these components.

The development of the team started in 2003 and a steady progress was observed since then. CAMBADA participated in several national and international competitions, including RoboCup world championships (5th place in 2007, 1st in 2008), the European "RoboLudens” and the annual Portuguese Open Robotics Festival (3rd place in 2006, 1st in 2007 and 2008). The good result obtained in RoboCup’2008 is largely due to the developed coordination methodologies, as the CAMBADA robots are among the slowest of the competition.

This paper is organized as follows: Section II presents the hardware and software architectures of CAMBADA players and provides details on the main software components involved in individual decisions of the players. Section III describes how players share information with teammates and how they integrate shared information. Section IV describes the adopted coordination methodologies. Section V presents and discusses competition results and various performance measures. Section VI concludes the paper.

II. PLAYER ARCHITECTURE

A. Hardware Architecture

The CAMBADA robots (Fig. 1) were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter. The mechanical structure of the players is

---

Nuno Lau and Luís Seabra Lopes are with the Transverse Activity on Intelligent Robotics of IEETA research unit as well as with the Department of Electronics, Telecommunications and Informatics, Universidade de Aveiro, Portugal. (e-mails: { nunolau, lsl }@ua.pt).

Gustavo Corrente was a researcher with the Transverse Activity on Intelligent Robotics of IEETA research unit, Universidade de Aveiro, Portugal, and is currently with Nokia Siemens Networks Portugal, Aveiro, Portugal. (e-mail: gustavo@ua.pt)

Nelson Filipe is an MSc student on Computer and Telematics Engineering at Universidade de Aveiro, Portugal (e-mail: nelson.filipe@ua.pt)

layered and modular. Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromechanical kicker, are attached to an aluminum plate placed 8 cm above the floor. The second layer contains the control electronics. The third layer contains a laptop computer, at 22.5 cm from the floor, an omni-directional vision system, a frontal camera and an electronic compass, all close to the maximum height of 80cm.

The players are capable of holonomic motion, based on three omni-directional roller wheels. With the current motion system, the robots can move at a maximum speed of 2.0 m/s. As mentioned, this is less that many of the other MSL teams, which can currently move at speeds typically between 2.5 and 4.0 m/s (e.g. [18] [10] [24] [6]).

The mentioned vision system allows detecting objects (ball, players, goals) and field lines on a radius of nearly 5m around each player. The frontal camera allows to detect the ball further away. Besides vision, each player includes encoders, battery status sensors and, for detecting if the ball is kickable, an infra-red presence sensor.

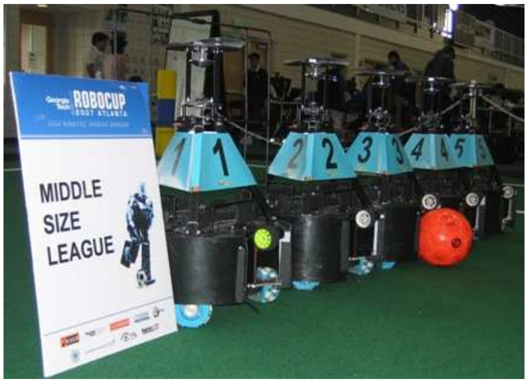


Fig. 1 CAMBADA robotic team

The robots computing system architecture follows the fine-grained distributed model [12] where most of the elementary functions, e.g. closed loop control of complex actuators, are encapsulated in small microcontroller based nodes, connected through a network. A laptop node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. USB or Firewire (Fig. 2). For this purpose, Controller Area Network (CAN), a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [8].

In the middle-size league, inter-robot communication and communication between the team's base station and the robots is extremely necessary for the team to maintain a coordinated behavior. The communication among robots and to the base station uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment. The base station is connected to the referee box through a wired LAN TCP link.

B. Software Architecture

The software system in each player is distributed among the various computational units (Fig. 2). High-level functions run on the computer, a laptop PC running Linux operating system. Low-level functions run partly on dedicated microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

A software module called the Integrator is used to update the world state information. This is done by filtering the raw information coming from sensors (i.e. vision, odometry, etc.) and determining the best estimate of the position and velocity of each object. Self-localization uses a sensor fusion engine based on the publicly available engine described in [14]. By integrating information from field line detection, this engine produces self position estimates with a high level of confidence. Compass information is used to resolve ambiguities.

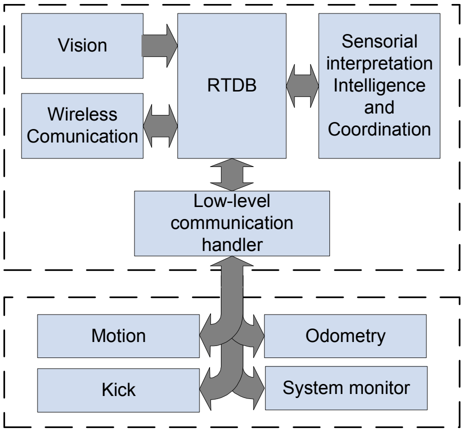

Vision
RTDB
Sensorial interpretation Intelligence and Coordination
Wireless Comunication
Low-level communication handler
Motion
Odometry
Kick
System monitor

Fig. 2. Layered software architecture of CAMBADA players, from [3]

The high-level processing loop starts by integrating perception information gathered locally by the player. This includes information coming from the vision processes, which is stored in a Local Area of the RTDB, and odometry information coming from the holonomic base via FTT-CAN. After integration, part of the world state is written in the shared area of the RTDB to make it available to teammates. The next step is to integrate local information with information shared by teammates.

The software of CAMBADA players is composed of several different processes that have responsibility 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]. Pman stores in a database the characteristics of each process to activate and allows the activation of recurrent tasks, settling phase control (through the definition of temporal offsets), precedence restrictions, priorities, etc. The Pman services allow changes in the temporal characteristics of the process schedule during run-time.

It is very important that all robots share the same play mode obtained by processing the referee orders given through the referee box. In CAMBADA, an application inside the team's base station checks the messages received from the "referee box", and converts the event triggered protocol of communication "referee box" - "base station" to a state oriented playmode information 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 minimized, enabling a synchronized collective behavior.

C. Roles and Behaviors

The CAMBADA player decision module is based on the concepts of *role* and *behavior*. Behaviors are the basic sensorimotor skills of the robot, like moving to a specific position or kicking the ball, while roles select the active behavior at each time step.

All roles are derived from a *Role* abstract class. Its *determineNextState()* method is responsible for the selection of the active behavior. To develop a new role, a *Role* derived class is created and the *determineNextState()* method is implemented. The *run()* method is implemented only in the base class and is responsible for the selection of the active behavior, using *determineNextState()*, and for its execution. To change the active behavior, the method *changeBehaviour(Behaviour*)*, implemented in the *Role* base class, is used.

During open play, the CAMBADA agents use only three roles: *RoleGoalie*, *RoleSupporter* and *RoleStriker*. The *RoleGoalie* is activated for the goalkeeper. *RoleSupporter* moves according to its strategic positioning. *RoleStriker* is an "active player" role. Other roles (*RoleBarrier, RoleReplacer, RoleToucher*) are used in set-pieces like kick-off, throw-in, goal-kick, corner-kick and free-kick. Further details about the developed roles and respective coordination mechanisms will be presented in section IV.

Behaviors are also organized as derived classes from an abstract class *Behavior*. It has three important methods. The first one is *calculate()* whose implementation, in derived classes, determines the parameter values of the low-level command that this behavior will execute (linear and angular velocities and kicking parameters). The second method is *execute()*, which sends previously computed parameters to the low-level computation modules. The separation of calculation and execution enables the agent to reason on the expected result of the commands while deciding which one to execute. Finally, *grabberControll()* controls the grabber mechanism automatically.

The set of behaviors that are implemented in the CAMBADA agent are adapted to its catadioptric omnidirectional vision and holonomic driving systems. The combination of these technologies enhances the set of possible behaviors when compared to a differential drive robot or to an holonomic drive robot with a limited field of view.

The behavior *bMove* uses two symbolic parameters: the target position where to move; and the position which the CAMBADA player should be facing in its path to the target. The symbols used are *OBall*, *TheirGoal* and *OurGoal*. The other moving behavior, *bMoveToAbs*, allows the movement of the player to an absolute position in the game field, and also allows the player to face any given position. Those moving behaviors may activate the functions of avoiding obstacles and avoiding the ball (used during the game repositions to avoid collisions with the ball). The *bPassiveInter* behavior moves the player to the closest point in the ball trajectory and waits there for the ball. The *bDribble* behavior is used to dribble the ball to a given relative player direction. The *bCatchBall* behavior is used to receive a pass. The player aligns itself with the ball path and, when the ball is close, moves backwards to soften the impact and more easily engage the ball. The *bKick* behavior is used to kick the ball accurately to one 3D position, either for shooting to goal or passing to a teammate. Preparing for the kick involves determining the kick direction and power. Polynomial functions, whose coefficients were determined by experimentation, are used to compute kick power based on distance to target. Different functions are used according to the expected number of ball bounces, given the distance. Finally, *bGoalieDefend* is the main behavior of the goalie.

III. INFORMATION SHARING AND INTEGRATION

Sharing perceptional information in a team can improve the accuracy of world models [5]. Sharing internal state can improve the team coordination. Therefore, information sharing and integration is one of the key aspects in multi-robot teams.

In CAMBADA, each robot uses some of the perceptions of the other robots, obtained through the RTDB, to improve its knowledge about the current positions and velocities of the other robots and of the ball. It is very important for our coordination model to keep an accurate estimate of the absolute position of the ball by each robot. The role assignment algorithm is based on the absolute positions of the robot and of its teammates. The teammates' positions are not obtained through the vision system and rely completely on the communicated estimated self positions of others.

Each agent communicates its own absolute position and velocity to all teammates as well as its ball information (position, velocity, visibility and engagement in robot), current role and current behavior.

The sharing of own absolute position and velocity is needed first of all because the vision system of the agents currently cannot detect the localization of the teammates. The vision system only detects obstacles but it doesn't try to detect individual robots within the detected obstacles nor does it try to determine if they are teammates or opponents. The absolute position of teammates is necessary to the strategy of our team, as the information is used to define our formation/strategy. So each robot trusts the estimated self positions of teammates that are communicated through the RTDB.

Multi-robot ball position integration has been used in the middle-size league by several teams [27][7]. In CAMBADA, multi-robot ball position integration is used to maintain an updated estimate of the ball position, when the vision subsystem cannot detect the ball, and to validate robot's own ball position estimate, when the vision subsystem detects a ball.

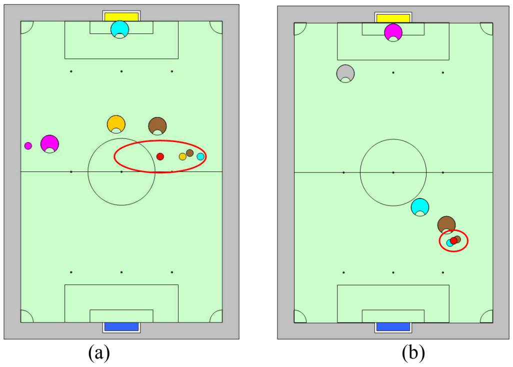

(a)
(b)

Fig. 3. Multi-robot ball position integration

Currently, a simple integration algorithm is used. When the agent doesn't see the ball, it analyzes the ball information of playing teammates. The analysis consists in the calculation of the mean and standard deviation of the ball positions, then discarding the values considered as outliers of ball position, and finally using the ball information of the teammate that has a shorter distance to the ball. To determine if the agent sees a fake ball, i.e., to validate the robot's own perception, we use a similar algorithm.

Communication is also used to convey the coordination status of each robot allowing robots to detect uncoordinated behavior, for example, several robots with the same exclusive role, and to correct this situation reinforcing the reliability of coordination algorithms.

The communication between the base station and the robots informs the robots of the active play mode (decided by the referee). During development, the base station can be used to control several robotic agent characteristics like fixed roles, manually activated self-positioning, etc, all managed through the RTDB.

IV. TEAM COORDINATION

The coordination model of the CAMBADA team is based on notions like *strategic positioning*, *role* and *formation*. These notions and related algorithms have been introduced and/or extensively explored in the RoboCup Soccer Simulation League [25][21]. In order to apply such algorithms in the Middle-Size League, several changes had to be introduced. The approach is presented in detail in this section.

A. Formations and strategic positionings

A formation defines a movement model for the robotic players. Formations are sets of strategic positionings, where each *positioning* is a movement model for a specific player. The assignment of players to specific positionings is dynamic, and is done according to some rules described below. Each positioning is specified by three elements:

- Home position, which is the target position of the player when the ball is at the centre of the field
- Region of the field where the player can move, and
- Ball attraction parameters, used to compute the *target position* of the player in each moment based on the current ball position

All these items of information are given in a strategy configuration file. Using different home positions and attraction parameters for the positionings allows a simple definition of defensive, wing, midfielder and attack strategic movement models. Fig. 4 shows the formation of the team used in RoboCup'2008 for several ball positions.

The definition of formation in terms of strategic positionings was introduced in the SBSP model [21] for the Soccer Simulation League. This model also introduced specific notions of *tactic* and *strategy*, which are currently not used in CAMBADA.

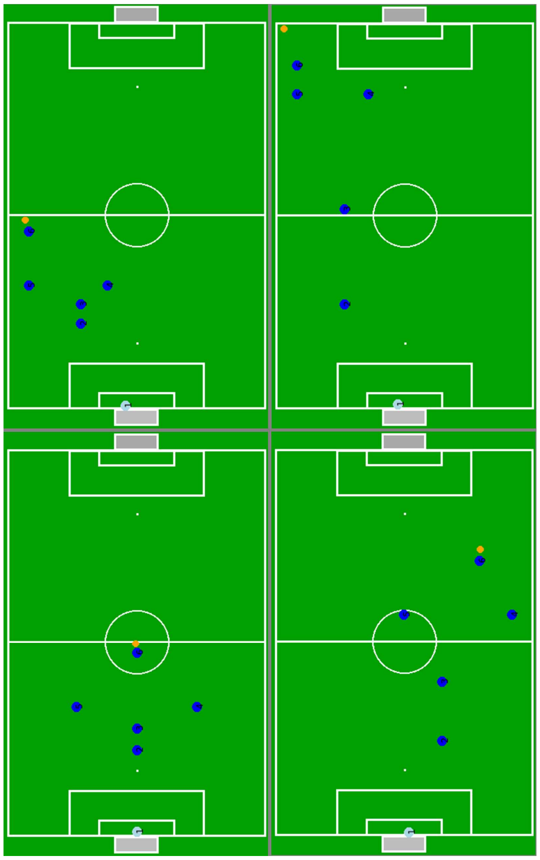


Fig. 4. Target player positions for several different ball positions

B. Roles in open play

As mentioned before, the CAMBADA players use only three roles in play-on mode: *RoleGoalie*, activated for the goalkeeper, *RoleSupporter* and *RoleStriker*. *RoleStriker* is an "active player" role. It tries to catch the ball and score goals. The striker activates several behaviors that try to engage the ball (*bMove*, *bMoveToAbs*), get into the opponent's side avoiding obstacles (*bDribble*) and shoot to the goal (*bKick*). The *bKick* behavior can perform 180º turns while keeping possession of the ball.

In a consistent role assignment, only one player at a time takes on the role of striker. The striker is helped by other teammates which take on *RoleSupporter* [13]. Supporters maintain their target positions as determined by their current positioning assignments and the current ball position. As a result, supporters accompany the striker as it plays along the field, without interfering. In case the ball is captured by the opponent, some supporter hopefully will be in a good position to become the new striker. Occasionally, supporters can take a more active behavior. This happens when the striker can't progress with the ball towards the opponent goal and, instead, the ball remains behind the striker for more than some pre-defined time (e.g. 2 seconds in the adopted configuration). In this case, the closest supporter to the ball also approaches the ball, in some sense acting as "backup striker".

C. *Role and positioning assignment*

The play-on decision that assigns roles and positionings to the active players is performed using an algorithm resembling DPRE [22], but with the innovation of considering different priorities for the different roles and positionings, so that the most important ones are always covered. This is an important feature since the number of available players varies as a result of several common situations in the MSL, namely hardware and software malfunctions and referee orders.

The algorithm is presented in Fig. 5. Consider a formation with N positionings and a team of $K \leq N$ available field players (not counting the goal-keeper which has a different mechanical configuration and therefore a fixed role). To assign the role and positioning to each robot, the distances of each of the robots to each of the target positions are calculated.

Then the striker role is assigned to the robot that is closest to the highest priority strategic positioning, which is in turn the closest to the ball. From the remaining K-1 robots, the closest to the defensive positioning (second highest priority) is assigned to this positioning, then the closest to the third level priority positioning is assigned next and the algorithm continues until all active robots have positionings and roles assigned. This algorithm results in the striker role having top priority, followed by the defensive positioning, followed by the other supporter positionings. The assignment algorithm may be performed by the coach agent in the base station, ensuring a coordinated assignment result, or locally by each robot, in which case the inconsistencies of world models may lead to unsynchronized assignments. In the latest competitions, positioning assignments were carried out by the coach at intervals of 1 second and the role assignments were individually carried out by each player.

D. *Passes*

Passing is a coordinated behavior involving two players, in which one kicks the ball towards the other, so that the other can continue with the ball. 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 [15] [26]. The CoPS team also support pass play [28].

```
Algorithm: role and positioning assignment
Input:
  POS - array of N positionings
  BallPos - ball position
Input/output:
  PL - array of K active players (K =< N)
Local:
  TP - array of N target positions
{
  clearAssigments(PL);
  TP = calcTargetPositions(POS,BallPos);
  for each POS[i], i ∈ 1..N, in
    descending order of priority
  {
    if there is no free player
      then return;
    p = the free player closest to TP[i];
    PL[p].positioning = i;
    PL[p].targetPosition = TP[i];
    if POS[i] has highest priority
      then PL[p].role = striker;
      else PL[p].role = supporter;
  }
  return;
}
```

Fig. 5. CAMBADA Positioning and role assignment algorithm

Two player roles have recently been developed for coordinated passes in the CAMBADA team. In the general case, the player running *RoleStriker* may decide to take on *RolePasser*, choosing the player to receive the ball. After being notified, the second player takes on the *RoleReceiver*.

These roles have not been used yet for open play in international competition games, but they have been demonstrated in RoboCup'2008 MSL *Free Technical Challenge* and a similar mechanism has been used for corner kicks (see below). In the challenge, two robots alternately took on the roles of passer and receiver until one of them was in a position to score a goal. The sequence of actions on both players is described in Table I. They start from their own side of the field and, after each pass, the passer moves forward in the field, then becoming the receiver of the next pass. The coordination between passer and receiver is based on passing flags, one for each player, which can take the following values: READY, TRYING_TO_PASS and BALL_PASSED. In the case of a normal game, another pass coordination variable would identify the receiver.

Table I – Coordinated actions in a pass

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

E. Set plays

Another methodology implemented in CAMBADA is the use of coordinated procedures for set plays, i.e. situations when the ball is introduced in open play after a stoppage, such as kick-off, throw-in, corner kick, free kick and goal kick. Set play procedures define a sequence of behaviors for several robots in a coordinated way. For that purpose, the involved players take on a specific roles.

RoleToucher and *RoleReplacer* are used to overcome the indirect rule in the case of indirect set pieces against the opponent. The purpose of *RoleToucher* is to touch the ball and leave it to the *RoleReplacer* player. The replacer handles the ball only after it has been touched by the toucher. This scheme allows the replacer to score a direct goal if the opportunity arises.

Two toucher-replacer procedures are implemented. In the case of *corner kicks*, the toucher passes the ball to the replacer and the replacer continues with the ball (see pseudo-code in Fig. 6). The passing algorithm is as explained above.

```
Algorithm: RoleReplacer // for corner kicks
{
  if I have Ball
    then shoot to opponent goal
    else if Ball close to me
      then move to Ball
      else
        if Toucher already passed ball
          then catch Ball
          else wait that Ball is passed
}
```

Fig. 6. Replacer role algorithm for corner kicks

Another toucher-replacer procedure is used in the case of *throw-in*, *goal kick* and *free kick* set plays. Here, the toucher approaches and touches the ball pushing it towards the replacer until the ball is engaged by the replacer, then withdraws leaving the ball to the replacer. The replacer also moves towards the ball, grabs it, waits that the toucher moves away and then shoots to the opponent goal. It should be noted that both the toucher and the replacer position themselves on the shoot line, so that, as soon as the toucher moves away, the replacer is ready to shoot. For the kick-off, a similar procedure is followed, but without reference to the shoot line, since the involved robots must be in their own side of the field.

Finally, in the case of set pieces against CAMBADA, *RoleBarrier* is used to protect the goal from a direct shoot. The line connecting the ball to the own goal defines the barrier positions. One player places itself on this line, as close to the ball as it is allowed. Two players place themselves near the penalty area. One player is placed near the ball, 45º degrees from the mentioned line, so that it can observe the ball coming into play and report that to teammates. Finally, one player positions itself in such a way that it can oppose to the progression of the ball through the closest side of the field. The placement of players is illustrated in Fig. 7.

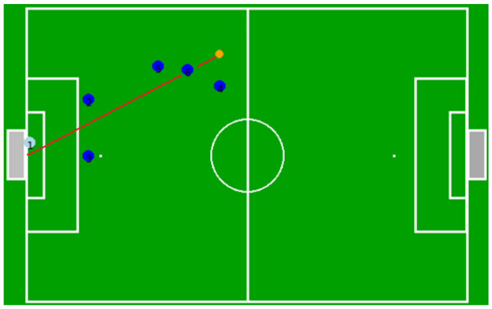


Fig. 7. Placement of RoleBarrier players

The assignment of the *RoleBarrier*, *RoleReplacer* and *RoleToucher* roles is executed by sorting the agents according to their perceived distances to the ball and selecting the closest ones, up to the maximum number of agents in each role. When selecting a role like the *RoleReplacer*, which is exclusive, the agent looks at the other teammates role decisions and if it finds a *RoleReplacer* with a lower uniform number it will never select that role. A similar approach is performed for the other exclusive roles. This assignment is always performed locally by each robot.

V. PERFORMANCE EVALUATION

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 analyzing log files and videos of games in this championship. In addition, RoboCup'2008 competition results will also be presented.

As the CAMBADA team made it to the final, it was scheduled to play 13 games. One of them was not played due to absence of the opponent. For two other games, the log files were lost. The results presented in the following are therefore extracted from log files of 10 games.

Table II 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. It can be seen that players spend a considerable amount of time in set plays (44% of the total time of the player in a game, including the *RoleParking*, which moves the player to a position outside the field at the end of the first half and at the end of the game). This reflects the current contingencies of MSL games. More time is spent in set plays against CAMBADA (28.4%, since usually four players take the barrier role in these situations) than in set plays against the opponent (11.5% in toucher and replacer roles). 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, all inconsistencies in the assignment of the striker role have a combined total duration of 20.9 ± 27.4 seconds in a game (~30 minutes). The high standard deviation results mainly from one game in which, due to magnetic interference, localization errors were higher than normal. In that specific game, role inconsistency occurred 45 times for a total of 101 seconds.

Table II – Average time spent by players in different roles (in %) and respective standard deviation

Role	%time
Striker	10.4 ± 5.2
Supporter	45.2 ± 10.0
Toucher	5.9 ± 4.1
Replacer	5.6 ± 4.6
Barrier	28.4 ± 6.5
Parking	4.4 ± 6.4

Table II shows the average percentage of time any given player spends running each implemented behavior. 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 highlight clearly 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.

Table III – Average time spent by players running different behaviors (in %) and respective standard deviation

Behavior	%time (any role)	%time (striker)
bMove	4.9 ± 3.0	43.7 ± 4.4
bMoveToAbs	74.7 ± 12.6	25.3 ± 4.7
bDribble	1.4 ± 1.2	13.4 ± 4.5
bKick	1.8 ± 1.5	14.6 ± 7.7
bCatchBall	0.2 ± 0.3	
bPassiveInter	0.3 ± 0.2	2.9 ± 1.1
bStopRobot	14.7 ± 7.0	

Concerning strategic positionings, relevant mainly to supporters as explained above, 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 9.829 ± 2.228 times per minute.

As the CAMBADA players to 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 IV. The closest player to the ball is at an average distance of 1.2 m from the ball (in a field of 18 m × 12 m). The ball is perceived by at least one player during 91.7% of the time. The ball is engaged in a player's grabber device 9.8% of the time.

Table IV – Measures related to ball possession

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

Fig. 8 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's side for 73% of time, and that the game was mainly being played in centre of the field, towards the opponent's side.

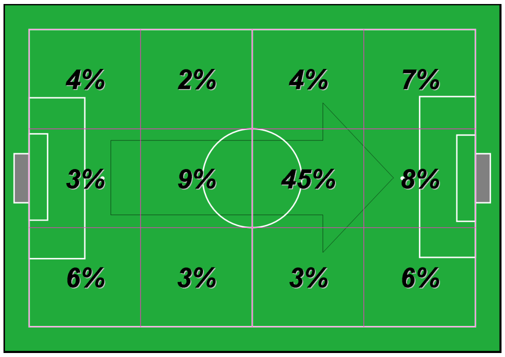


Fig. 8. Percentage of time the ball was in different locations of the field in 10 games (CAMBADA on the left)

Fig. 9 shows the location in the field from where the ball was shot to goal in the RoboCup'2008 final (CAMBADA-TechUnited).

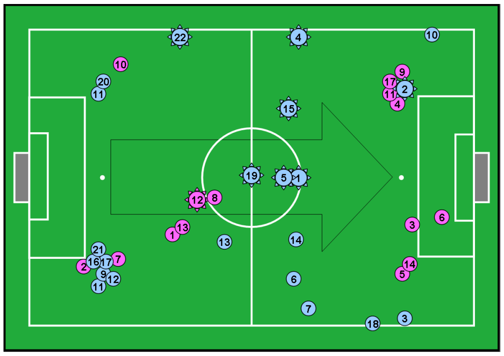


Fig. 9. Shoot locations in the final CAMBADA (left) - TechUnited (right) game in RoboCup 2008 (successful shoots are simple circles and goals are sun-like forms)

Table V 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.

Table V – Competition results

	#games	#goals 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

VI. CONCLUSION

The data structures used for world state representation clearly separate the raw sensor information from the world model that results of integrating local and shared information. This architecture is easily adaptable to the addition of new sensors.

The adaptation of SBSP and DPRE [22][21] to the Middle-Size League environment resulted in a coordinated behavior of the team that contributed to its recent successes. The robot malfunctions decrease the number of field players, but the positioning/role assignment algorithm maintains a competitive formation with fewer players in the field. Set plays were very efficient as several of the CAMBADA goals were the direct result of their activation.

Following steady progress since the start of development of CAMBADA in 2003, the team won the RoboCup'2008 championship. This good result is largely due to the developed coordination methodologies, as the CAMBADA robots are among the slowest of the competition.

ACKNOWLEDGMENT

The CAMBADA team was funded by Portuguese Government – FCT, POSI/ROBO/ 43908/2002 (CAMBADA) and currently FCT, PTDC/EIA/70695/2006 (ACORD).

REFERENCES

[1] Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes, Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences -- ISCIS 2004: 19th International Symposium, Proceedings*, Aykanat, Cevdet; Dayar, Tugrul; Korpeoglu, Ibrahim, eds., Lecture Notes in Computer Science, Vol. 3280, 2004, pp. 876-886.

[2] Arbatzat, M., et al.: Creating a Robot Soccer Team from Scratch: the Brainstormers Tribots, *Proceedings of Robocup 2003, Padua*, Italy, 2003.

[3] Azevedo, J.L., M.B. Cunha, L. Almeida, Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. Proc. *ETFA2007- 12th IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, 2007, pp. 973-980.

[4] Cunha, B., J. Azevedo, N. Lau, L. Almeida, Obtaining the Inverse Distance Map from a Non-SVP Hyperbolic Catadioptric Robotic Vision System, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*, LNAI, Springer Verlag, Berlin, 2008.

[5] Dietl, M., J.-S. Gutmann and B. Nebel, Cooperative Sensing in Dynamic Environments, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, Maui, Hawaii.

[6] *EtherCAT Robots win German Open*, Press Release, EtherCAT Technology Group, 8 May 2008, at http://ethercat.org/pdf/english/etg_032008.pdf

[7] Ferrein, A., L. Hermanns and G. Lakemeyer, Comparing Sensor Fusion Techniques for Ball Position Estimation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, 2006, pp. 154-165.

[8] Ferreira, J.; Pedreiras, P.; Almeida, L.; Fonseca, J.A. The FTT-CAN protocol for flexibility in safety-critical systems, *IEEE Micro*, 22 (4), 2002, pp. 46-55.

[9] Figueiredo, J., Lau, N., Pereira, A. Multi-Agent Debugging and monitoring framework, *Proc. First IFAC Workshop on Multivehicle Systems (MVS'06)*, Brasil, October, 2006.

[10] Hafner, R., S. Lange, M. Lauer, and M. Riedmiller (2008) Brainstormers Tribots Team Description, *RoboCup International Symposium 2008, CD Proceedings*, Suzhou, China.

[11] Kok, J.; Spaan, M. and Vlassis, N., Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50 (2-3), Elsevier Science, 2005, pp. 99-114.

[12] Kopetz, H., *Real-Time Systems Design Principles for Distributed Embedded Applications*, Kluwer, 1997.

[13] Lau, N., L., Seabra Lopes, G. Corrente (2008) CAMBADA: Information Sharing and Team Coordination, Autonomous Robot Systems and Competitions: Proceedings of the Eigth Conference. 2 April 2008, Aveiro, Portugal, Universidade de Aveiro, p. 27-32.

[14] Lauer, M., S. Lange and M. Riedmiller, Calculating the perfect match: An efficient and accurate approach for robot self-localisation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., LNCS 4020, Springer, 2006.

[15] Lima, P, L. Custódio, I. Akin, A. Jacoff, G. Kraezschmar, B. Kiat Ng, O. Obst, T. Röfer, Y. Takahashi, C. Zhou (2005) RoboCup 2004 Competitions and Symposium: A Small Kick for Robots, a Giant Score for Science, *AI-Magazine*, 6 (2), 2005, p. 36-61.

[16] MSL Technical Committee 1997-2008, *Middle Size Robot League Rules and Regulations for 2008. Draft Version - 12.2 20071109*, November 9, 2007.

[17] Neves, A.; Corrente, G. and Pinho A., An omnidirectional vision system for soccer robots. *Progress in Artificial Intelligence*, Lecture Notes in Computer Science. Berlin, nº 4874, Springer, 2007, pp. 499-507.

[18] Oubbati, M., M. Schanz, T. Buchheim, P. Levi (2006) Velocity Control of an Omnidirectional RoboCup Player with Recurrent Neural Networks, A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Computer Science, vol. 4020, 691-701.

[19] Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup-2005: Robot Soccer World Cup IX*, I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, Berlin, 2006, pp. 371-383.

[20] Pedreiras, P.; Almeida, L., Task Management for Soft Real-Time Applications Based on General Purpose Operating Systems, *Robotic Soccer*, edited by: Pedro Lima, Itech Education and Publishing, Vienna, Austria, 2007, pp. 598-607.

[21] Reis, L.P. and N. Lau, and E.C. Oliveira, Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents, *Balancing Reactivity and Social Deliberation in Multiagent Sytems: From RoboCup to Real Word Applications*, M. Hannenbauer, J. Wendler, and E. Pagello eds., LNAI 2103, Springer-Verlag, 2001, pp. 175-197.

[22] Reis, L.P. and N. Lau, FC Portugal Team Description: RoboCup 2000 Simulation League Champion, *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, et al. eds., LNCS 2019, Springer, 2001, pp. 29-40.

[23] Riedmiller, M. Gabel, T., On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup, *Proceedings of the 3rd IEEE Symposium on Computational Intelligence and Games (CIG 2007)*. IEEE Press, April 2007, pp. 17-23.

[24] 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 (2008) Hibikino-Musashi Team Description Paper, RoboCup International Symposium 2008, CD Proceedings, Suzhou, China.

[25] Stone, P. and M. Veloso, Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork, *Artificial Intelligence*, vol. 110 (2), 1999, pp. 241-273.

[26] van der Vecht, B., and P. Lima (2005) Formulation and Implementation of Relational Behaviours for Multi-robot Cooperative Systems, RoboCup 2004: Robot Soccer World Cup VIII, Springer LNCS Volume 3276/2005, pp 516-523.

[27] Weigel, T. W. Auerbach, M. Dietl, B. Dümler, J.S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski and M. Thiel, CS Freiburg: Doing the Right Thing in a Group, *RoboCup 2000: Robot Soccer World Cup IV*, P. Stone, G. Kraetzschmar, T. Balch, eds., Springer-Verlag, 2001, pp. 52-63.

[28] Zweigle, O., R. Lafrenz, T. Buchheim, U.-P. Käppeler, H. Rajaie, F. Schreiber and P. (2006) Levi Cooperative agent behavior based on special interaction nets, *Intelligent Autonomous Systems 9*, T. Arai, R. Pfeifer, T. Balch, and H. Yokoi, Eds. Amsterdam, The Netherlands: IOS Press.