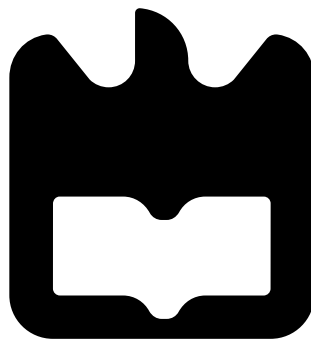




**Sérgio Filipe
Silva Martins**

Treinador automático para a equipa CAMBADA





**Sérgio Filipe
Silva Martins**

Treinador automático para a equipa CAMBADA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Nuno Lau (orientador), Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor António Neves (coorientador), Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

júri / the jury

presidente / president

Prof. Doutor Luís Filipe de Seabra Lopes

Professor Associado da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Paulo José Cerqueira Gomes da Costa

Professor Auxiliar da Universidade do Porto (Arguente Principal)

Prof. Doutor António José Ribeiro Neves

Professor Auxiliar da Universidade de Aveiro (Coorientador)

agradecimentos / acknowledgements

Os meus primeiros agradecimentos vão para os meus pais. Sem eles seria impossível chegar até aqui pois a pessoa que sou hoje a eles o devo sem a menor dúvida. Devo ao meu pai a curiosidade, a procura da descoberta, a necessidade prática, o culto do ócio e uma inumerável quantidade de valias que só tenho a agradecer. À minha mãe devo a persistência, o afinco, o suor, a esperança e tudo o que precisa uma pessoa quando os objetivos a alcançar são árduos. Aos dois um muito obrigado pois tenho muito orgulho em vós e penso que sentem o mesmo em relação ao vosso filho.

Não posso deixar de agradecer ao meu irmão. Quem poderia um viajante querer para companheiro de viagem se não um outro viajante com o mesmo destino. Pois bem, apesar de teres estado comigo na universidade apenas um ano foi sem dúvida o melhor. Rafael, obrigado pela paciência que tiveste nas minhas manhãs menos boas, pelas fantásticas mãos de cozinheiro que vieram dar alegria à nossa mesa e principalmente pela companhia e camaradagem que partilhamos este último ano. Os manos não se escolhem mas para mim só me interessa que não mudem.

Querida Susana, prometo compensar-te por todos os fins de semana que ficaste sem mim. A tua boa disposição foi muito importante, já para não falar das horas que me aturaste a falar desta minha dissertação. Para ti, um beijo grande e um obrigado ainda maior.

Muitos amigos vieram e muitos foram, porém existem amigos que nos aparecem de malas e bagagem, amigos de que nunca nos iremos esquecer. Os amigos marcam-nos em alturas não programadas e sem aviso prévio. Ao Luís, meu amigo desde que me conheço. Ao Crispim, esse meu grande professor da velha arte do manejo do ferro de soldar. Ao Stoyanov, o meu primeiro e grande amigo de universidade. Ao Pardal, embaixador da noite e exímio animal de palco. Ao José Rocha e à Sónia Rocha, eles que me ajudaram a desvendar os mistérios da negra e sombria matemática. Ao Flávio, imagem da boa formalidade ao vivo e a cores. Ao Pinto, um patrão muito atencioso com os seus pedaços de terra. Todos eles deixaram uma assinatura na minha formação como estudante e como pessoa. Com o mesmo cariz incondicional da vossa amizade deixo-vos aqui o meu grande obrigado por tudo.

À Engenheira Advogada Carla Monteiro mais conhecida por Madrinha Carla. Podia ter uma estrela de cinema como exemplo a seguir mas para isso a madrinha também teria de se mudar para Hollywood. A imagem que me passou foi sempre determinante nas minhas decisões. Todas as pessoas deviam ter uma Madrinha Carla.

Por último mas sem menos louvor quero agradecer aos meus orientadores, ao João Silva e ao Ricardo Dias por todo o apoio, ajuda, conhecimento e prontidão com que me guiaram neste projeto. Sem a intervenção destas pessoas teria sido muito mais difícil e demorado terminar com sucesso este trabalho e fazer surtir os resultados que conseguimos.

Resumo

A robótica é a área da ciência que estuda o *design*, construção e uso de máquinas para executar tarefas tradicionalmente feitas por seres humanos. O futebol não é apenas o desporto rei da humanidade mas também o desporto mais usado para o desenvolvimento de robôs com capacidades desportivas.

A iniciativa RoboCup promove competições robóticas a nível mundial, sendo o futebol robótico uma das mais importantes, contando com participantes de todo o mundo. A equipa CAMBADA da Universidade de Aveiro trabalha desde 2003 na construção de robôs que competem na chamada liga dos robôs médios do RoboCup.

Este trabalho descreve o desenvolvimento dum treinador automático para a equipa CAMBADA. Com este novo agente, a equipa consegue agora que as suas ações em campo sejam mais versáteis, controlando de forma dinâmica as posições dos robôs, isto é, a sua formação em jogo, fazendo uso de regras que definem qual a formação a usar. As regras são criadas com base na informação que os robôs conhecem sobre o estado do jogo, como por exemplo posição de bola, tempo de jogo e golos. Utilizando um ambiente gráfico criado para configurar as regras do novo treinador temos uma ferramenta completa e funcional.

O interesse demonstrado pelas outras equipas no desafio científico do evento RoboCup 2013, onde a equipa CAMBADA apresentou o treinador descrito nesta dissertação, permitiu obter um importante 3º lugar. Em competição, este treinador foi também utilizado ajudando a equipa CAMBADA a alcançar o 3º lugar no campeonato mundial de futebol robótico - RoboCup 2013 (liga dos robôs médios).

Abstract

Robotics is a field of science that studies the design, construction and use of machines to execute tasks traditionally executed by humans.

Soccer is not only the king of sports, but also the most used sport for developing robots with sport capabilities.

The RoboCup is an initiative that promotes worldwide robotics competitions, being one of the most important in the land soccer robotics. Since 2003, the CAMBADA team, from the University of Aveiro, develops and builds robots that compete in the RoboCup Soccer Middle Size League.

This thesis describes the development of an automatic coach for the CAMBADA team. This new agent enables the team to have more versatile actions on the field, dynamically controlling the positions of the robot, i.e., the team formation, using rules which define the formation to use. That rules are defined, based on the information each robot knows about the game state, such as ball position, playing time and goals. Using a graphical environment designed to configure the new coach rules, the team has a complete and functional tool.

The interest shown by other teams in the Scientific Challenge of RoboCup 2013, where CAMBADA team presented the coach described on this dissertation, led to an important third place in this challenge. This coach was also used by CAMBADA in this competition games, helping the team to reach the third place on the RoboCup 2013 MSL tournament.

A Match

"I build robots all day,
I have nothing more to say.
Poetry is simple and bliss,
the gears click, pneumatics hiss.

My machine is waiting for the match to start,
my pounding heart, it will not stop.
Poetry is really not my thing,
the match starts, the bell rings.

Five hundred lines of code execute at once,
all in a small jump.
At once my robot shines,
just like this poem, nothing but lines.

Two minutes later, the match is done.
It is over, we have won."

por Cody Smith

Conteúdo

Conteúdo	i
Lista de Figuras	iii
Lista de Tabelas	v
Acrónimos	vii
1 Introdução	1
1.1 Arquiteturas multiagente	2
1.2 RoboCup	4
1.2.1 RoboCup Soccer	4
1.2.2 RoboCup Rescue	7
1.2.3 RoboCup@Home	9
1.2.4 RoboCup Junior	9
1.2.5 RoboCup@Work	10
1.2.6 Logistics League	11
1.3 Objetivos e Contribuições	11
1.4 Estrutura	12
2 Estrutura do CAMBADA	13
2.1 Comunicação de baixo nível	13
2.2 RTDB	15
2.3 HWcomm	16
2.4 Base Station	16
2.5 Visão	17
2.6 Agente Robótico	17
2.7 Coach	18
2.8 Posicionamento Estratégico	18
3 Trabalho Desenvolvido	21
3.1 CoachConfig	21
3.1.1 Lista de Formações	22
3.1.2 Lista de Regras	22
Parâmetros Comuns	23
Tipos de Regras	23
Default Formation	25

3.1.3	Outros parâmetros de configuração	26
	<i>Away Fight Time</i> e <i>Fight Distance</i>	26
	<i>Window Distance Map Size</i>	26
3.1.4	Menus e Operações	27
	Interoperabilidade <i>CoachConfig/Coach</i>	27
	Configure Coach	28
3.2	Novo <i>Coach</i>	29
3.2.1	Funcionamento	31
3.2.2	Obtenção do dados de entrada	33
	<i>Flow</i>	33
	Mapa de distribuição da bola no campo	35
3.2.3	Avaliação das regras e propagação dos resultados	35
3.3	Alterações à <i>Base Station</i>	37
4	Resultados Experimentais	39
4.1	Desafio Científico	39
4.1.1	Apresentação do Desafio Científico	40
4.1.2	Classificação	43
5	Conclusões	45
A	Ficheiro CoachConfigFile.xsd	47
B	Ficheiro de configuração usado no desafio científico do Robocup 2013	49
C	Tabela de Classificações do Desafio Científico do RoboCup 2013	51
D	Fotografia das Classificações do Desafio Científico do RoboCup 2013	53
E	Slides da apresentação do novo <i>Coach</i> no Desafio Científico do RoboCup 2013	55
	Bibliografia	59

Lista de Figuras

1.1	Projeto SARTRE - Veículos ligeiros coloridos de verde formam um pelotão [22].	2
1.2	Comunidades de agentes que seguem metodologias MAS.	4
1.3	Robôs da <i>Standart Platform League</i> [7].	5
1.4	Robôs da <i>Humanoid League</i> [6].	5
1.5	Robôs da <i>Small Size League</i> da equipa FU-Fighters(2004-2005) [50].	6
1.6	Robôs da <i>Middle Size League</i> . Equipa CAMBADA no Róbotica 2011 [3].	6
1.7	<i>Screenshot</i> da <i>Simulation League 2D</i> [52].	7
1.8	Robô Hector da liga <i>Rescue Robot</i> (German Open 2010) [51].	8
1.9	Resultado do mapa Instambul3 da equipa <i>GUC ArtSapience</i> na final da liga <i>Rescue Simulation</i> no RoboCup 2013 [13].	8
1.10	Robô <i>Dynamaid</i> da Universidade de Bonn (<i>RoboCup @Home</i> 2009) [8].	9
1.11	Robôs da liga <i>Junior Soccer</i> [18].	10
1.12	Robôs da liga <i>Junior Dance</i> [16].	10
1.13	Robô da liga <i>Junior Rescue</i> [17].	10
1.14	Robô da liga <i>RoboCup@Work</i> [12].	11
1.15	Prova do RoboCup 2012 no México da <i>Festo Logistics Competition</i> [24].	11
2.1	Ligação dos vários componentes de <i>software</i> à RTDB.	14
2.2	Comunicação entre os módulos de <i>hardware</i> e a camada de <i>software</i> na equipa CAMBADA.	14
2.3	Uso da zona local e partilhada da RTDB no robô 1 e os seus intervenientes.	15
2.4	Interface disponibilizada pela <i>Base Station</i>	17
2.5	Interface de configuração dos <i>Setpieces</i>	20
2.6	Interface de configuração das Formações DT.	20
3.1	CoachConfig - Lista de formações usada no desafio científico do Robocup 2013.	22
3.2	CoachConfig - Lista de regras usada no desafio científico do Robocup 2013.	23
3.3	Efeito de <i>threshold</i> na ativação duma regra <i>Our Field Time</i>	25
3.4	Treinador automático no CAMBADA - Leitura e escrita de ficheiros.	28
3.5	Coach - <i>Interface</i> de amostragem de distribuição da bola pelo campo.	30
3.6	Coach - Tabela de amostragem do resultado da avaliação das regras usadas no desafio científico do Robocup 2013.	30
3.7	Coach - Fluxo da iteração principal.	32
3.8	Coach - Fluxo da função <i>UpdateStatistics</i>	34
3.9	Coach - Fluxo da função <i>Evaluate Rule</i>	36

3.10	Base Station - Modo de formação automático permitindo visualizar qual a formação DT em vigor.	37
3.11	Base Station - Modo de formação manual permitindo selecionar qual a formação DT em vigor.	37
4.1	Desafio Científico - Formação A	41
4.2	Desafio Científico - Formação B	41
4.3	Desafio Científico - Formação Default	42

Lista de Tabelas

3.1	<i>Context Value</i> em função do tipo de regra	24
4.1	Classificação final do Desafio Científico do RoboCup 2013	43

Acrónimos

AGV	<i>Automated Guided Vehicle</i>
CAMBADA	C ooperative A utonomous M obile r o B ots with A dvanced D istributed A rchitecture
DT	<i>Delaunay Triangulation</i>
CAN	<i>Controller Area Network</i>
FIFA	<i>Fédération Internationale de Football Association</i>
MAS	<i>Multi-Agent System</i>
MSL	<i>Midle Size League</i>
NASA	<i>National Aeronautics and Space Administration</i>
SARTRE	<i>Safe Road Trains for the Environment</i>
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
IMU	<i>Inertial Measurement Unit</i>
HWcomm	<i>Hardware Comunication Process</i>
RTDB	<i>Real Time Database</i>
XML	<i>eXtensible Markup Language</i>
XSD	<i>XML Schema Definition</i>
FSM	<i>Flow State Machine</i>

Capítulo 1

Introdução

Desde muito cedo que o homem tende a formar grupos para realizar tarefas. A grande capacidade de comunicação entre indivíduos que o homem desenvolveu ao longo dos tempos foi para isto fulcral, tendo feito com que o trabalho em grupo se tenha vindo a estender a quase todas as atividades da raça humana. Esta forma de comportamento tem acompanhado evoluções da própria raça atravessando grandes eras da história permanecendo até aos dias de hoje, vejamos: o homem pré-histórico planeava ataques em grupo aos animais selvagens para obter alimento em abundância [32], o Faraó Quéops levantou o seu túmulo, entre outras situações históricas, com uma mão-de-obra que se estima em cerca de 100 mil pessoas ao longo de 20 anos [38]. Na atualidade poucos trabalhos são feitos por uma pessoa singular, a maioria das pessoas trabalha numa empresa onde cada uma tem as suas tarefas e responsabilidades definidas, isto leva a que as equipas precisem de comunicar entre si para sincronizar e maximizar o esforço a fim de cumprir uma tarefa maior.

Nos desafios propostos ao nível tecnológico é também cada vez mais comum as tarefas serem realizadas não apenas por um elemento, mas sim por vários elementos computacionais por vezes designados por agentes. Podemos então definir um agente como sendo uma entidade computacional, situada num determinado ambiente, que recolhe informação do mundo que a rodeia através de sensores, que toma decisões e atua de forma autónoma por meio de atuadores, especificamente desenhados a fim de cumprir os objetivos para os quais foram projetados [37]. Num sistema onde vários agentes cooperam entre si pressupõe-se a troca de informação, a fim de atingir um comportamento global, inteligente e a uma escala maior, tirando partido dum conhecimento mais alargado [37].

A agência *National Aeronautics and Space Administration* (NASA) há já alguns anos que estuda como tornar os seus satélites mais autónomos a fim de aumentar as suas capacidades de decisão [39]. O projeto *Safe Road Trains for the Environment* (SARTRE) está a tentar revolucionar o transporte pessoal trabalhando num sistema que visa agrupar veículos automóveis em pelotões [30] (Figura 1.1) por forma a que estes se movimentem sem a intervenção do condutor. Para implementar este sistema está a ser ponderado dotar veículos pesados conduzidos por condutores profissionais como líderes duma fila de veículos, todos os veículos à retaguarda assumem a liderança para o veículo que os antecede criando uma corrente de veículos que não precisarão da intervenção do condutor para os conduzir. Cada veículo tem ainda a obrigação de se proteger de forma autónoma garantindo assim a segurança dos seus ocupantes mesmo que os outros agentes funcionem de forma incorreta ou algo não previsto aconteça. Em funcionamento existem já alguns sistemas que trabalham em equipa. No ramo dos transportes

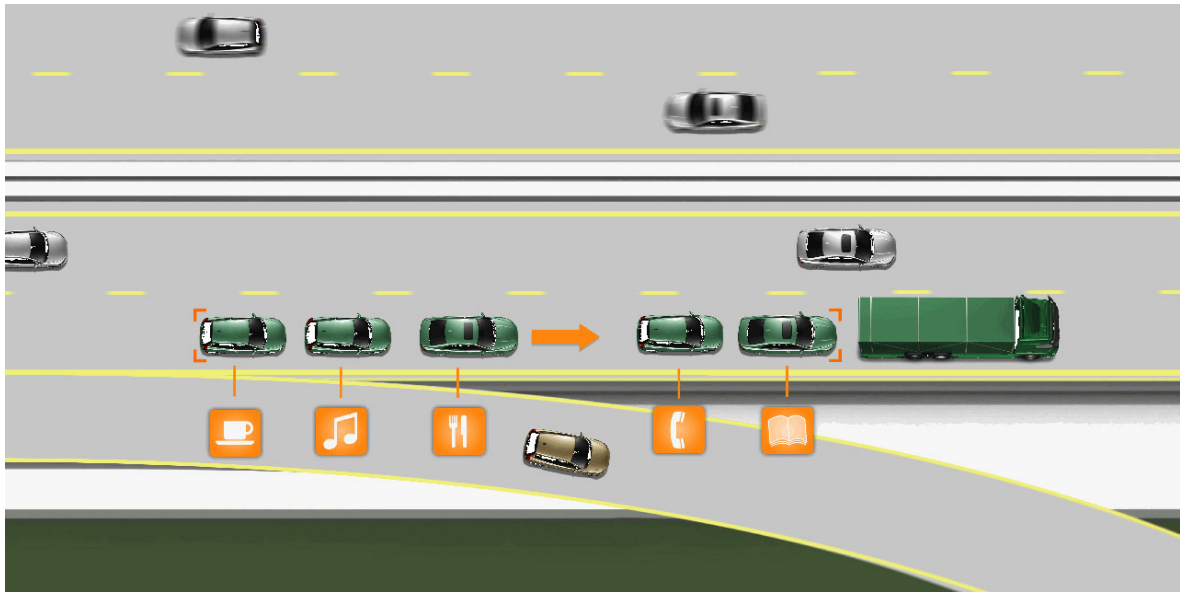


Figura 1.1: Projeto SARTRE - Veículos ligeiros coloridos de verde formam um pelotão [22].

de mercadorias a empresa americana TEREX[®] disponibiliza já como produto aos seus clientes um sistema *Automated Guided Vehicles* (AGVs) para transportar contentores horizontais entre os cais de carga e zona de pilhagem de contentores [2].

1.1 Arquiteturas multiagente

As soluções computacionais distribuídas têm vindo a crescer de interesse nos últimos anos. A causa deste aumento prende-se geralmente com dois grupos diferentes de motivações:

- **Soluções de problemas distribuídas** - Geralmente este tipo de soluções prende-se com a natureza do problema. Ao se subdividir o problema em partes, com algum grau de independência, obtêm-se implementações mais simples. Também quando o problema infere uma carga computacional elevada recorre-se ao uso destes sistemas, conseguindo-se assim com que sistemas computacionais com diferentes capacidades e diferentes localizações geográficas possam contribuir para a resolução do problema [31].
- **Multi-Agent Systems (MASs)** - Focadas na intercomunicação entre agentes, cada um executa as suas tarefas que podem ou não ser diferentes entre agentes diferentes. Neste tipo de sistemas cada interveniente é dotado da sua própria Inteligência Artificial (IA) [29] que lhe confere autonomia. Por outro lado, através da partilha de informação entre todos os intervenientes, estes agem e decidem com base em Inteligência Artificial Distribuída (IAD) [29,37].

Na elaboração dum sistema MAS é necessário pensar no agente como entidade singular como também nas capacidades de todos os agentes ao agirem como grupo. Desta forma permitirá aos agentes cooperarem, coordenarem e negociarem entre si, mesmo que os seus interesses ou objetivos individuais sejam diferentes. Por vezes a natureza de alguns problemas faz com que as suas soluções tenham de passar por sistemas multiagente, um dos exemplos

disso mesmo é o reconhecimento e monitorização de áreas de grande dimensão. Se tal tarefa fosse executada por uma entidade singular apenas teríamos a monitorização da área adjacente ao mesmo, se mais agentes se distribuírem pela área teremos uma visão muito mais alargada em tempo real.

Quanto mais autónomo for um agente mais robusto este se torna, mesmo quando todos os outros agentes falham este pode continuar as suas tarefas que sejam independentes dos restantes, ou que por falta de IAD sejam feitas de forma menos eficiente. Outra vantagem do uso de MAS é a possibilidade de dividir as características dos agentes. Por exemplo no Brasil, a plantação das novas colheitas de soja é feita logo de seguida às colheitas atuais por duas categorias de máquinas distintas, estas cooperam entre si durante as suas tarefas. Caso as tarefas não fossem divididas por duas máquinas diferentes teríamos apenas uma máquina de porte bastante maior, de complexidade acrescida para abarcar com mais funcionalidades e como consequência de tudo isso um custo muito mais elevado.

A interação entre vários agentes deve ter em consideração a forma como vão comunicar. Não é desejado que tarefas com restrições temporais sejam executadas tardiamente porque os agentes passaram demasiado tempo a trocar informações. Também deve ser sempre tomado em conta o impacto que tem a falha dum agente no desempenhar duma tarefa. Quando o sistema prevê ações mecânicas e perceção do mundo real surgem dificuldades acrescidas. Estas vêm com a tolerância dos sensores, variação do meio ambiente e falta de exatidão das ações mecânicas já que nem sempre o meio é bem comportado e totalmente previsto.

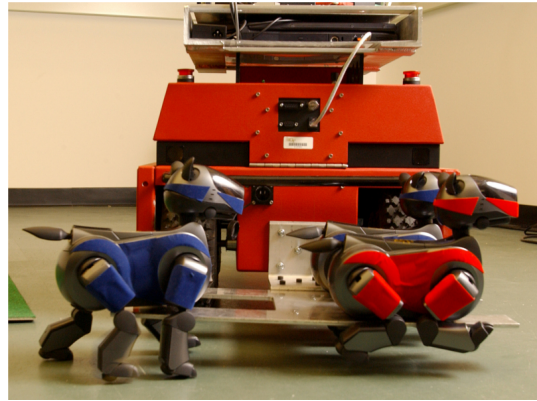
Os sistemas MAS podem ser classificados da seguinte forma [34]:

- **Homogéneo** - Todos os agentes têm a mesma construção, os mesmos objetivos, o mesmo tipo de conhecimento e a mesma capacidade de agir. Organizam-se entre si e decidem as suas ações com base nas no mundo por eles percebido e também com base nas informações oferecidas pelos outros agentes.
- **Heterogéneo** - O caso oposto ao descrito anteriormente em que os agentes têm que cooperar por forma a maximizar as diferentes capacidades de cada um em específico.
- **Enxames** - Baseado em comunidades existentes na natureza e muitas vezes usado para o estudo e análise das mesmas, tais como insetos e formigas, em que um número elevado de agentes de capacidades muitas reduzidas (Figura 1.2a) conseguem executar tarefas de forma cooperativa através de IAD.
- **Marsupial** - Como o próprio nome sugere prevê-se a existência dois tipos de agentes que diferem das suas capacidades e funcionalidades por terem objetivos diferentes e relacionarem-se entre si de forma hierárquica (Figura 1.2b). Os agentes da primeira camada têm geralmente objetivos de maternidade sustentando e fornecendo recursos aos agentes da camada seguinte que tenta completar as tarefas do problema.

Existem muitas áreas onde se utiliza MAS para tentar resolver problemas de grande complexidade. O futebol robótico é uma área de investigação científica bastante complexa onde esta metodologia pode ser testada na maioria dos seus objetivos. Os investigadores que se dedicam à resolução deste problema enfrentam vários desafios, tais como: perceção do mundo, localização, necessidade de trabalho em equipa, tarefas com restrições temporais, possibilidade dos agentes terem capacidades diferentes, necessidade de controlar os tempos de comunicação, necessidade de concretizar algoritmos de elevada complexidade, nomeadamente no âmbito do tratamento de imagem, objetivos dinâmicos dependendo do estado do jogo,



(a) Robô Jasmine - Robôs Homogêneos organizados em enxame [53].



(b) BORG USAR - Rôbos Marsupiais Heterogêneos [23].

Figura 1.2: Comunidades de agentes que seguem metodologias MAS.

entre outros. Desta forma, e com praticamente todos os problemas a incidir na ordem de trabalhos apontada pela metodologia, o futebol robótico é um problema ideal para testar e trabalhar formas de resolver problemas complexos em MAS.

1.2 RoboCup

A iniciativa RoboCup surgiu em 1997 sendo que a primeira edição teve lugar no Japão na cidade de Nagoya. Um dos objetivos é promover a conceção duma equipa de futebol robótico que em 2050 seja capaz de ganhar à equipa humana campeã do mundo sob as regras da *Fédération Internationale de Football Association* (FIFA). O RoboCup conta anualmente com um evento a nível mundial e vários eventos regionais [15]. Este apresenta-se com um formato atraente para o grande público sendo aberto a toda a comunidade. O evento promove o desenvolvimento da robótica de ponta com um nível de competição elevado entre os participantes, sendo que a grande maioria dos participantes são estudantes de vários graus de escolaridade. A diversidade de modalidades que compõem a competição tem vindo a expandir-se, no primeiro evento era composto apenas por três ligas de futebol robótico, um desafio científico e um desafio tecnológico. A sua última edição teve lugar na Holanda na cidade de Eindhoven [14] e contou com as seguintes modalidades:

1.2.1 RoboCup Soccer

A *RoboCup Soccer* é a vertente mais diversificada no RoboCup. Sempre tentando criar um ambiente de futebol robótico, o *RoboCup Soccer* conta com várias modalidades. As modalidades variam em muitos aspetos: escala dos agentes, liberdade nas técnicas de construção, formas de comunicação, ambiente real ou virtual, número de jogadores e faixa etária dos participantes, entre outros.

- **Standart Platform League** - todas as equipas competem usando humanoides H25 NAO fabricados pela Aldebaran Robotics. Nesta modalidade os robôs são completamente autónomos comunicando entre si e recebendo o estado do jogo via *wireless* [20].

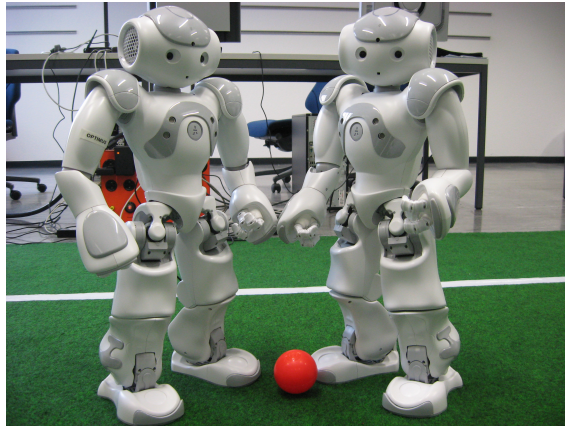


Figura 1.3: Robôs da *Standart Platform League* [7].

- **Humanoid League** - Nesta modalidade é focada as capacidades sensoriais dos humanos, sendo que apenas são permitidos sensores que operem de forma semelhante aos sistemas sensoriais dos seus criadores. Da mesma forma apenas são permitidos atuadores que se assemelhem a ações humanas, como exemplo temos que uso de mecanismos como pernas robóticas para a locomoção do robô são obrigatórios impossibilitando o uso de rodas com tração [43].



Figura 1.4: Robôs da *Humanoid League* [6].

- **Small Size League** - As equipas são formadas por robôs de dimensões reduzidas, sem sistema de visão próprio e recebendo ordens via rede sem fios. O campo dispõe de duas câmaras elevadas que identificam todos os objetos em campo, comunicando às equipas os dados pré-processados. As equipas, processam e decidem como se devem comportar os seus jogadores em campo informando-os sobre o que fazer. Esta competição explora a computação centralizada com atuação multiagente distribuída [47].



Figura 1.5: Robôs da *Small Size League* da equipa FU-Fighters(2004-2005) [50].

- ***Midle Size League (MSL)*** - Todos os tipos de atuadores e sensores são permitidos desde que não comprometam o funcionamento das equipas adversárias. As restrições são mais genéricas focando as suas regras para questões como o tamanho dos robôs, peso, manutenção do canal de transmissão partilhado, drible natural da bola e obrigatoriedade relativamente ao jogo em equipa. A forma como as equipas equipam os seus robôs é perfeitamente livre. Desta forma verificamos jogos entre equipas totalmente diferentes tanto a nível tecnológico como a nível estratégico. A competição é renhida e todos os anos as equipas se tornam mais capazes trazendo inovações a cada evento. Conta também com dois desafios: desafio técnico e desafio científico. No desafio técnico as equipas demonstram as suas capacidades como equipa mobilizando os seus robôs para demonstrações de cooperação. No desafio científico cada equipa demonstra um desafio que enfrentou e como o resolveu sendo este avaliado pelas restantes equipas, é neste desafio que geralmente as equipas mostram as suas linhas de investigação na área [44].



Figura 1.6: Robôs da *Middle Size League*. Equipa CAMBADA no Róbotica 2011 [3].

- ***Simulation Leagues*** - Nestas ligas nenhum *hardware* físico é mantido pois os sensores dos jogadores são simulados ficando assim em foco a inteligência artificial requerida para

um jogo de futebol. A liga de simulação 2D é jogada à semelhança dum jogo tradicional de futebol num plano. É um jogo muito fluido pois o ambiente é muito bem previsto e o controlo dos jogadores é bastante simplificado. Já na liga de simulação 3D, onde os robôs são modelados em 3D, mais variáveis para além da adição da terceira dimensão importam para o jogo, um exemplo disso passa pelo controlo dos robôs mais sofisticado contando com noções de equilíbrio, inercia, centros de massa e colisões [48].

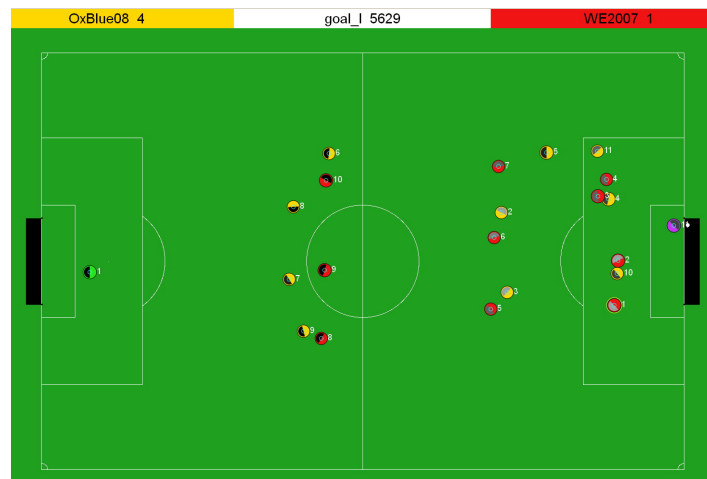


Figura 1.7: Screenshot da *Simulation League 2D* [52].

1.2.2 RoboCup Rescue

- **Rescue Robot** - Esta prova insere-se no âmbito de busca e salvamento em cenários de catástrofe. Vários objetivos são esperados à medida que as equipas promovem investigação nesta área: coordenação multiagente, reconhecimento de terreno acidentado, assistência de primeiros socorros às vítimas, simulação de cenários e sistema de apoio à decisão, teste e análise de estratégias de salvamento e visão para sistemas futuros. Em competição os robôs são inseridos num ambiente hostil criado para o efeito que tenta recriar um edifício que tenha sido alvo de catástrofe, tendo 20 minutos para fazer o reconhecimento da área devastada e encontrar o maior número possível de vítimas, devem executar procedimentos de primeiros socorros e servindo de meio de comunicação entre os socorristas e a vítima [46].

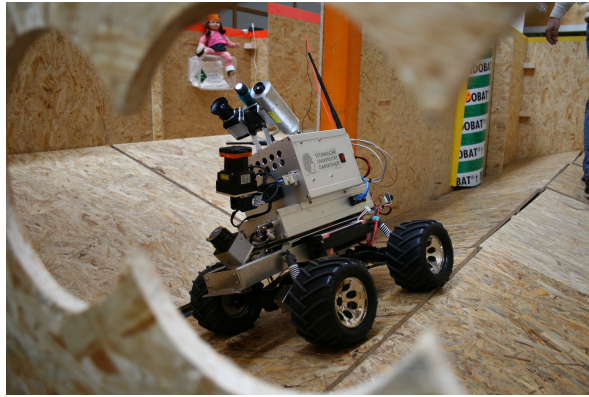


Figura 1.8: Robô Hector da liga *Rescue Robot* (German Open 2010) [51].

- ***Rescue Simulation*** - Esta prova prevê um cenário mais alargado do que a anterior concentrando-se na coordenação entre agentes multidisciplinares tais como bombeiros, ambulâncias e policias. A simulação pode ser seguida com interfaces que tentam também elas simular os meios disponíveis em ambientes reais como fotos aéreas conseguidas por um helicóptero (Figura 1.9). A simulação não se limita a um edifício mas sim numa cidade em apuros onde a eficácia da utilização dos recursos é fundamental para salvar vidas. A avaliação das equipas é feita em vários momentos da simulação da catástrofe, sempre comparando os resultados da equipa com os resultados sem intervenção [45].

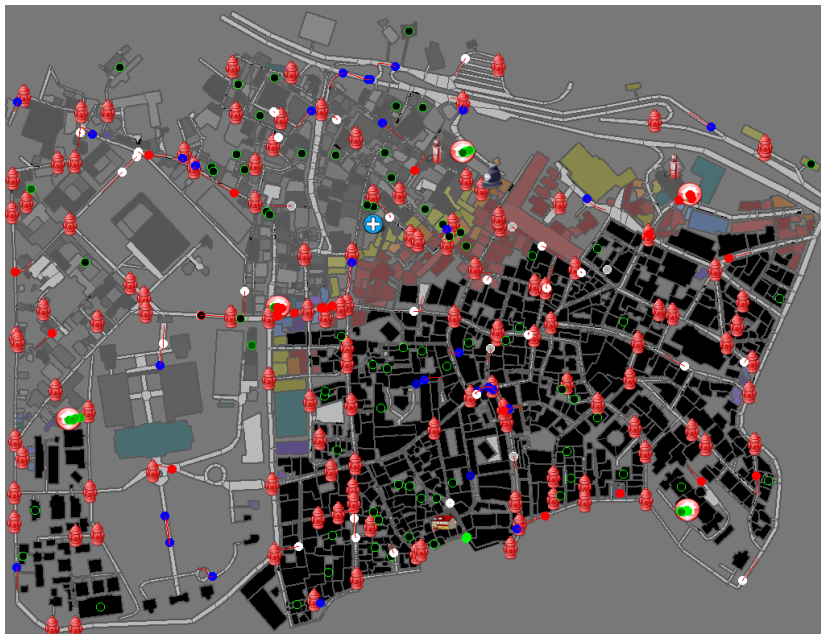


Figura 1.9: Resultado do mapa Istanbul3 da equipa *GUC ArtSapience* na final da liga *Rescue Simulation* no RoboCup 2013 [13].

1.2.3 RoboCup@Home

Robôs de aspecto amigável realizam tarefas domésticas com o objetivo de facilitar o cotidiano das pessoas ou ajudar pessoas em condições de cuidados acrescidos. À medida que as equipas vão ficando mais capazes os desafios vão-se tornando mais complexos em prol do avanço da área. Os robôs são colocados não só em ambientes domésticos como em outros ambientes onde as pessoas precisam de ir com frequência, o exemplo disso mesmo é uma ida ao supermercado fazer as compras. Assim os companheiros robóticos têm que passar por testes que necessitam de auto-localização, reconhecimento de espaços dinâmicos, reconhecimento de pessoas e objetos interagindo com os mesmos [42].

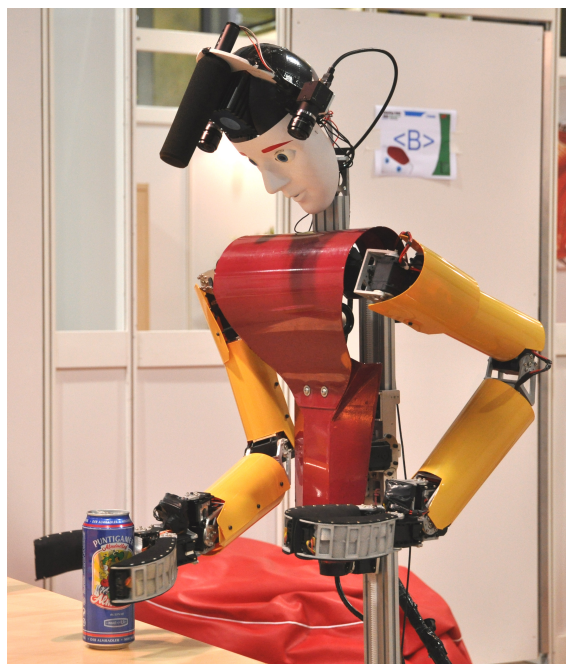


Figura 1.10: Robô *Dynamaid* da Universidade de Bonn (*RoboCup @Home* 2009) [8].

1.2.4 RoboCup Junior

Este projeto é pensado para os mais novos oferecendo-lhes uma experiência científica internacional, tendo nesta liga a possibilidade de estes darem os seus primeiros passos na robótica. Existem três categorias onde os mais novos fazem as suas primeiras experiências:

- **Junior Soccer** - Os estudantes têm de conceber uma equipa de dois robôs ao nível do *hardware* e do software. Contra uma equipa adversária e seguindo uma bola emissora de infravermelhos tentam ganhar um jogo de futebol [21].

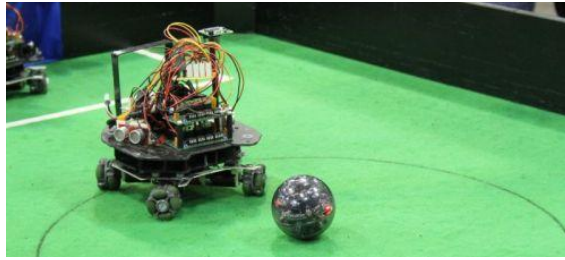


Figura 1.11: Robôs da liga *Junior Soccer* [18].

- *Junior Dance* - Devidamente ornamentados, como se de dançarinos se tratassem, um ou mais robôs executam danças sincronizadas elaboradas previamente pelas equipas [5].



Figura 1.12: Robôs da liga *Junior Dance* [16].

- *Junior Rescue* - As equipas preparam os seus agentes de salvamento para seguir linhas por vezes acidentadas a fim de encontrar vítimas numa situação de busca e salvamento [11].

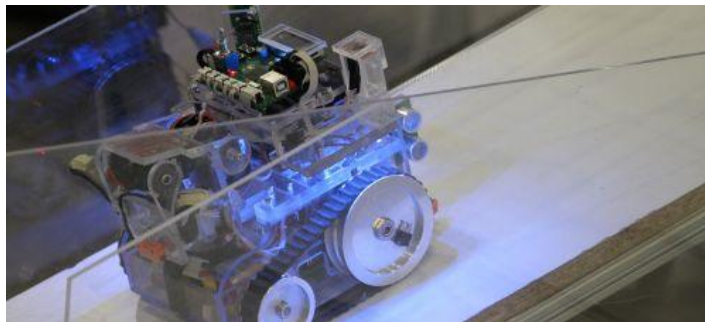


Figura 1.13: Robô da liga *Junior Rescue* [17].

1.2.5 RoboCup@Work

Esta é uma liga recente com problemas semelhantes à *RoboCup@Home* em que os objetivos passam pela logística e tarefas em ambientes industriais.



Figura 1.14: Robô da liga *RoboCup@Work* [12].

1.2.6 Logistics League

Esta ligam, baseada na plataforma robótica Robotino[®] da Festo, investiga problemas logísticos preparando-se para a próxima era de agentes AGV. Três robôs autônomos com capacidade de comunicarem entre si tentam resolver problemas logísticos não conhecidos pelas equipas no início de cada prova [19].

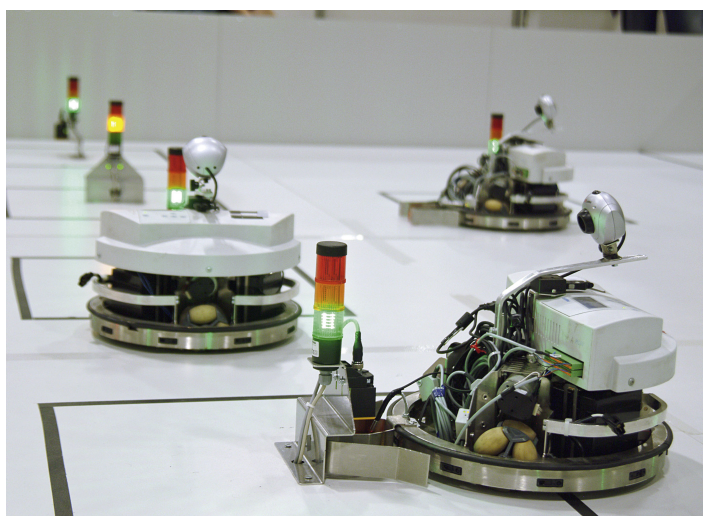


Figura 1.15: Prova do RoboCup 2012 no México da *Festo Logistics Competition* [24].

1.3 Objetivos e Contribuições

Este documento descreve o desenvolvimento dum treinador automático para equipas de futebol robótico. A equipa **Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture (CAMBADA)** do grupo de investigação em robótica da Universidade de Aveiro será o ponto de partida para os testes e implementação da proposta.

A tarefa deste novo agente será selecionar uma formação de jogo com base em regras pré-estabelecidas. As regras terão de ser definidas antes dos momentos de jogo, por forma a não ser necessária a intervenção humana durante o período de confronto das equipas.

Terão de ser tomadas em conta situações de falha, pois uma falha neste novo agente não poderá comprometer o comportamento dos jogadores em campo prejudicando assim o jogo.

O trabalho desenvolvido consistiu na definição de um novo modelo de treinador, através da implementação de um novo agente designado *Coach*, que utiliza um conjunto de regras para a escolha da formação a usar pela equipa num determinado momento de jogo. A configuração destas regras fica a cargo de uma ferramenta gráfica, designada por *CoachConfig*, que gera um ficheiro *eXtensible Markup Language* (XML) que será depois interpretado pelo *Coach*. Houve, também, a necessidade de alterar algumas ferramentas já existentes no projeto para que este novo treinador fosse usado.

No fim deste trabalho o novo *Coach* mostrou ser uma ferramenta estável e completa a fim de ser usada pela equipa numa competição. Isto comprovou-se no RoboCup 2013 onde a equipa CMBADA utilizou com sucesso este novo agente durante o torneio. Também de frisar que este trabalho foi avaliado pelas principais equipas mundiais de futebol robótico no desafio científico do RoboCup 2013.

1.4 Estrutura

No Capítulo 2 será introduzida a equipa CMBADA. Pretende-se assim, explicar o ponto da situação relativamente à organização do sistema computacional multiagente em questão, desenvolvido e levado a cabo nos últimos anos pelo grupo de investigação em robótica da Universidade de Aveiro.

No Capítulo 3 será retratado o trabalho realizado na criação do treinador automático proposto. Serão explicadas em detalhe as duas aplicações, *Coach* e *CoachConfig*, assim como as estruturas de dados por elas utilizadas e a forma como o utilizador as pode manipular. Também terão lugar neste Capítulo todas as alterações efetuadas no projeto CMBADA a fim de introduzir e utilizar este novo agente.

No Capítulo 4 será explicado de que forma foi explorado o problema em termos da experimentação. Ainda neste Capítulo será exposto o resultado da avaliação deste novo conceito no desafio científico no RoboCup 2013.

Por último, no Capítulo 5 serão feitas as considerações finais relativamente ao projeto e trabalho desenvolvido, terminando com o trabalho futuro que poderá ser desenvolvido no futuro.

Capítulo 2

Estrutura do CAMBADA

A CAMBADA, iniciada em Outubro de 2003 [1], é a equipa da liga MSL do RoboCup desenvolvida pela Universidade de Aveiro. A equipa compete em eventos nacionais e internacionais contando já com vários títulos de prestígio (RoboCup: 5^o lugar em 2007, 1^o lugar em 2008 e 3^o lugar em 2009, 2010, 2011 e 2013; 2^o lugar no German Open em 2010; 3^o lugar no Dutch Open em 2012; Festival de Robótica 2013: 3^o lugar em 2006, 1^o lugar em 2007, 2008, 2009, 2010, 2011 e 2012 e 2^o lugar em 2013). Para além das competições em campo a equipa tem conseguido boas qualificações nos *Technical Challenges* do RoboCup (2^o lugar em 2008, 1^o lugar em 2009, 4^o lugar em 2010, 1^o lugar em 2012 e 2013) assim como nos *Free Challenges/Scientific Challenges* (1^o lugar em 2011 e 2^o lugar em 2012 e 2013).

Professores e alunos da Universidade de Aveiro desenham e projetam desde a estrutura e componentes mecânicos até ao *hardware* e *software*. Trabalhos na área de reconhecimento de imagem, eletrónica de controlo e sistemas de computação distribuída veem no projeto CAMBADA a oportunidade de testar ideias e conceitos, desta forma professores e alunos canalizam para este projeto muita da investigação efetuada na universidade. A equipa, multidisciplinar, tem vindo a crescer contando já com mais de 30 colaboradores.

2.1 Comunicação de baixo nível

A arquitetura do CAMBADA baseia-se no paradigma biomórfico [36] composto por duas camadas: a camada de coordenação trata das operações de comunicação, operações de grande carga computacional e transferência de grandes quantidades de dados (Figura 2.1); a camada de baixo nível tem menor capacidade de transferência de dados e trata do controle dos dispositivos de *hardware* do robô (Figura 2.2).

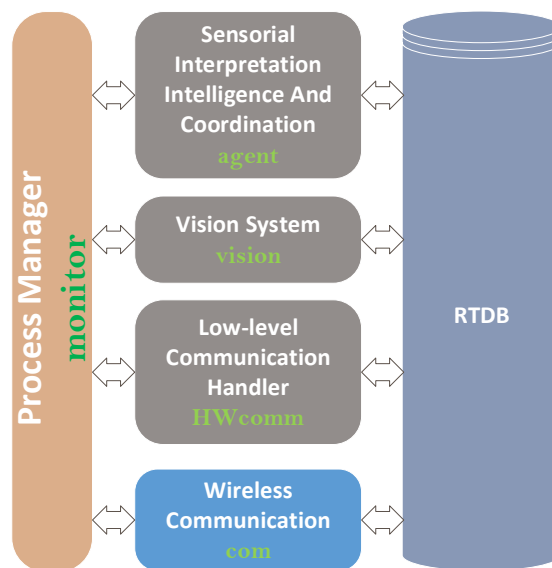


Figura 2.1: Ligação dos vários componentes de *software* à RTDB.

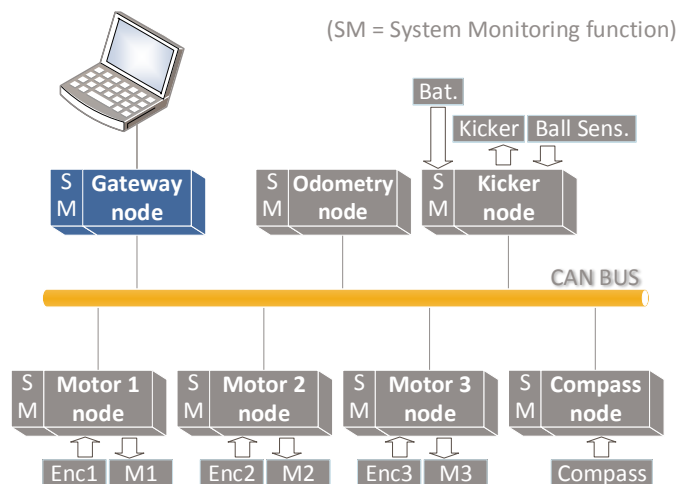


Figura 2.2: Comunicação entre os módulos de *hardware* e a camada de *software* na equipa CAMBADA.

Na camada de baixo nível residem as unidades mais elementares do robô. Cada unidade tem um conjunto de funções específicas e bem definidas: movimento, chute, odometria e monitorização. Para comunicar com estes dispositivos foi escolhida a tecnologia *Controller Area Network* (CAN) pela simplicidade de desenvolvimento de dispositivos que comuniquem desta forma.

O movimento holonómico dos robôs é conseguido com 3 motores DC espaçados em ângulos de 120° transmitindo força de tração a rodas do tipo *OmniWheels*. Este tipo de roda tem

tração num sentido permitindo o deslocamento lateral em simultâneo. Sistemas compostos por três rodas deste tipo são geralmente designados *Kiwi Drive* [28]. As funções de odometria, executadas pelo *odometry manager* que lê os dados dos *encoders* instalados nos 3 motores, combina esses mesmos dados e envia para o *gateway* o vetor de deslocamento do robô ($\Delta x, \Delta y, \Delta \theta$) [27]. Para efeitos de complementaridade às funções de odometria, os robôs têm instalada uma *Inertial Measurement Unit* (IMU). A IMU tem um acelerómetro tridimensional e um giroscópio incorporado que combinados permitem detetar falhas oriundas da odometria. É também neste módulo que reside uma bússola magnética. Esta, permite à equipa indicar a orientação do campo o que auxilia os robôs na tarefa de saber qual o seu meio campo.

O controlo da bola é composto por dois sistemas: *Kicker* e *Grabber*. O *Kicker* é formado por um atuador eletromecânico que atua na bola a fim de a chutar, permitindo ao robô efetuar tanto um passe rasteiro da bola como um remate em arco. O *Grabber* permite ao robô puxar a bola para si a fim de a poder manipular driblando-a. O *Grabber* tem atuadores mecânicos que exercem pequenas forças controladas sobre a bola na direção do robô. O sistema de monitorização controla os dados vitais do robô, tais como estado das baterias e estado dos restantes modelos funcionais.

2.2 RTDB

No projeto CAMBADA é dada uma grande importância à partilha de informação entre agentes. É a partir da partilha de informação que a equipa decide as suas tarefas e age de forma cooperativa [26]. O objetivo da existência duma *Real Time Database* (RTDB) é a sincronização dos processos inter-robô e intra-robô. A informação que tem origem no próprio robô é local e apenas os processos locais a manipulam. Para alcançar a sincronização inter-robô existe uma zona partilhada onde cada interveniente tem um *slot* sua e a difunde pelo canal de comunicação. A zona partilhada da RTDB em cada robô é o conjunto de todas as zonas partilhadas dos robôs (Figura 2.3).

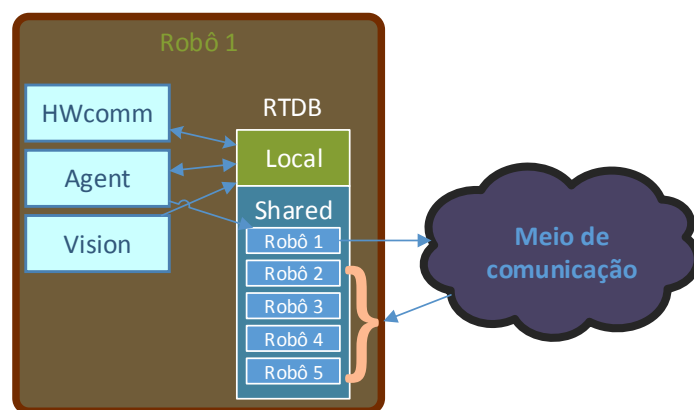


Figura 2.3: Uso da zona local e partilhada da RTDB no robô 1 e os seus intervenientes.

Para impedir que o canal de transmissão seja mal utilizado pelos agentes, cada agente tem um *time slot* definido para transmissão dos seus dados em *broadcast*. Geralmente o

mecanismo de *time sharing* da unidade de processamento em sistemas operativos de uso geral não é orientado a aplicações de tempo real, o que tornaria impossível sincronizar as transmissões em intervalos de tempo bem definidos. As informações partilhadas por cada robô têm um cariz temporal, que é calculado aquando dos pedidos locais à RTDB.

A replicação dos dados em todos os agentes serve maioritariamente para obter dados que o agente pode necessitar para decisão comportamental. Um exemplo disso mesmo é quando um agente precisa de localizar a bola e esta não se encontra no seu ângulo de visão. Neste tipo de situações o agente usa a informação disseminada pelos outros agentes para determinar a posição da bola, após ter esta informação este continua as suas tarefas.

2.3 HWcomm

O processo *Hardware Communication Process* (HWcomm) é o responsável pelas trocas de informação entre os módulos CAN do robô e os processos que estão a ser executados no computador. Usa a RTDB para obter a informação de *input* para os microcontroladores e também para depositar a informação de *output* proveniente dos dispositivos sensoriais.

2.4 Base Station

Um jogo na MSL é arbitrado por humanos, sendo que quando o árbitro intervém sobre o jogo as suas ações são comunicadas via *RefBox* [10] e disseminadas para as equipas. Desta forma, a *Base Station* [33], durante o decorrer dos jogos, serve de intermediário entre os agentes da equipa e a *RefBox*. Porém, o uso da *Base Station* por parte da equipa não se cinge ao momento de jogo. É na *Base Station* que reside um grande conjunto de ferramentas de teste, controlo e monitorização que são usadas pela equipa para desenvolvimento e testes.

A *Base Station* compreende um conjunto de funcionalidades de controlo alto nível relativamente aos agentes de campo. Usando a RTDB para difundir as ordens é possível controlar o comportamento dos agentes. As funcionalidades que existem desde a primeira versão oficial são: sinais de jogo (ex.: **Start** e **Stop**), indicação da cor da equipa (Azul Ciano ou Magenta), indicação da baliza (Azul Clara ou Amarela) e indicação de *Roles* [33].

Na liga MSL as equipas não podem interferir no jogo. Assim sendo, a *Base Station* permite monitorizar o jogo numa só janela, mostrando graficamente as informações que os agentes estão a disseminar na RTDB durante os jogos. Ao nível da monitorização a *Base Station* permite visualizar os seguintes parâmetros: estado do jogo, posição da bola e dos robôs, ângulo dos robôs, velocidade em plano da bola e dos robôs, estado das baterias dos robôs, *Roles* e comportamentos assumidos pelos agentes, visibilidade e posse de bola, cor atual da equipa, qual o meio campo da equipa e ainda pontos para efeitos de *debug* [33].

Dentro do projeto CMBADA este é um *software* crítico pois não é permitida a intervenção humana durante o decorrer dum jogo. Assim a *Base Station* tem de ser robusta, pois toda a equipa depende dela. O *design* gráfico tem de ser bastante cuidado possibilitando em jogo a monitorização da maior quantidade de informação possível sem intervenção humana. A Figura 2.4 mostra a interface principal desta aplicação.



Figura 2.4: Interface disponibilizada pela *Base Station*.

2.5 Visão

A visão é o principal subsistema de dados de *input* para o agente. Através da visão o agente recolhe informações que lhe permitem saber a sua localização, detetar a bola e detetar obstáculos existentes no campo.

A visão do CAMBADA baseia-se num sistema de visão omnidirecional catadióptrico [36]. Este sistema é constituído por uma câmara de vídeo digital apontada a um espelho hiperbólico. O robô com a função de guarda-redes tem uma câmara de perspetiva adicional para a deteção da bola num referencial 3D [36].

Sendo o processo com maior carga computacional para o agente, os tempos de processamento de cada uma das *frames* das câmaras têm de ser bem conhecidos e controlados.

2.6 Agente Robótico

O processo **Agente** é o responsável por interpretar os dados dos módulos sensoriais, tomar decisões comportamentais e trabalhar a coordenação com os outros robôs. Cada robô tem definido um identificador único que é lido pelo **Agente** quando este é iniciado. Uma vez atribuídos os identificadores o **Agente** que tiver o menor identificador assume a função de guarda-redes da equipa. Os restantes irão alternando as suas funções dependendo do estado do jogo, posse de bola e posição da mesma.

O processo principal do agente robótico é dividido nas seguintes fases: fusão sensorial, posicionamento estratégico, decisão e execução. Na fusão sensorial, levada a cabo pelo *Integrator*, é reunida a informação recolhida pelos sensores do robô e fundida com a informação partilhada pelos restantes colegas da equipa. Uma vez reunida a informação o *World State* é atualizado e será usado para decisão e coordenação de alto nível.

Na fase de posicionamento estratégico o robô tenta alcançar uma posição estratégica. Aqui nasce a noção de formação da equipa que irá ser explicada mais à frente na Secção 2.7.

Na fase de decisão o agente assume um *Role* o que confere ao robô a noção de atitude. A função de guarda-redes referida acima é um *Role* assim como o robô que mais próximo da bola se encontrar assume o *Role* de *Striker*. Quando a equipa está a tentar recuperar a posse de bola, o *Role* de *Striker* é sempre assumido pelo robô que se encontrar mais perto da bola. O robô que assume este *Role* persegue a bola tentando recupera-la. Quando um robô deixa de pertencer à formação, o que acontece no *Role Striker*, deixa desocupada a sua posição estratégica. Desta forma, e porque as posições estratégicas têm prioridades, os robôs de campo durante o jogo vão trocando não só de *Roles* mas também de posições estratégicas.

A fase de decisão opera sobre os atuadores de baixo nível, refletindo-se assim os comportamentos inerentes à atitude do agente. Estes comportamentos, denominados de *Behaviours*, depositam na RTDB as ações a serem executadas pelos atuadores de baixo nível que serão executadas pelos módulos de controlo do *hardware*.

2.7 Coach

O trabalho descrito nos Capítulos seguintes baseia-se no agente *Coach* previamente existente no CAMBADA. Este agente apenas tinha a seu cargo a atribuição dos posicionamentos estratégicos a cada jogador, garantindo desta forma um ponto de sincronismo entre as posições tomadas pelos robôs de campo [36]. Os algoritmos de atribuição dos posicionamentos estratégicos são conhecidos tanto pelo *Coach* como pelos jogadores. Quando o *Coach* não está disponível os jogadores calculam as suas posições estratégicas sob a pena poder ser atribuída a mesma posição estratégica a dois robôs em simultâneo.

2.8 Posicionamento Estratégico

Os conceitos do modelo de coordenação da equipa CAMBADA baseiam-se em posições estratégicas, papéis e formações [35]. Uma formação é um conjunto de posições estratégicas que definem um modelo de movimento para cada um dos jogadores em campo. A atribuição dum jogador a uma posição é feita dinamicamente por um conjunto de regras predefinidas [35].

Existem três tipos de formação usadas dependendo do contexto:

- Formações dos **Setpieces** [41] - Os **Setpieces** definem sequências de ações em lances de bola parada da própria equipa. São utilizadas formações para definir as posições dos jogadores intervenientes nas jogadas deste tipo criadas especificamente para cada **Setpiece**.
- **Formação em *Free Play*** - Formação usada em jogo aberto quando a bola está a ser disputada pelas duas equipas. Cada formação em *Free Play* define a posição no campo

a ser tomada por cada posicionamento estratégico tendo em conta a posição da bola ao longo de todo campo.

O as formações deste tipo são formações *Delaunay Triangulation* (DT). Este tipo de formação, baseado em fundamentos matemáticos propostos por Boris Delaunay em 1943 [49], está bem descrita em [25].

- **Barreiras** - Esta formação é uma Formação DT ativada apenas quando o adversário está a marcar um lance de bola parada. É muito útil visto que com apenas uma formação se obtêm todas as linhas de barreira para todo o campo.

Existe para cada tipo de formação uma ferramenta de edição adequada para tornar mais fácil a sua alteração. Sendo que temos uma interface para os *Setpieces* (Figura 2.5) e outra para as Formações DTs tanto de jogo aberto como das Barreiras (Figura 2.6).

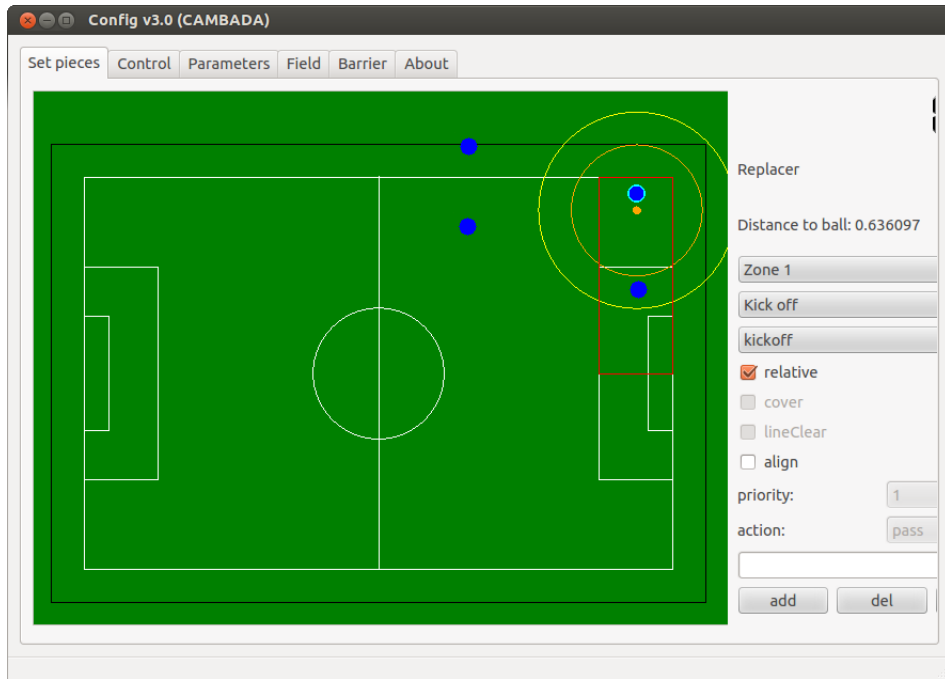


Figura 2.5: Interface de configuração dos Setpieces.

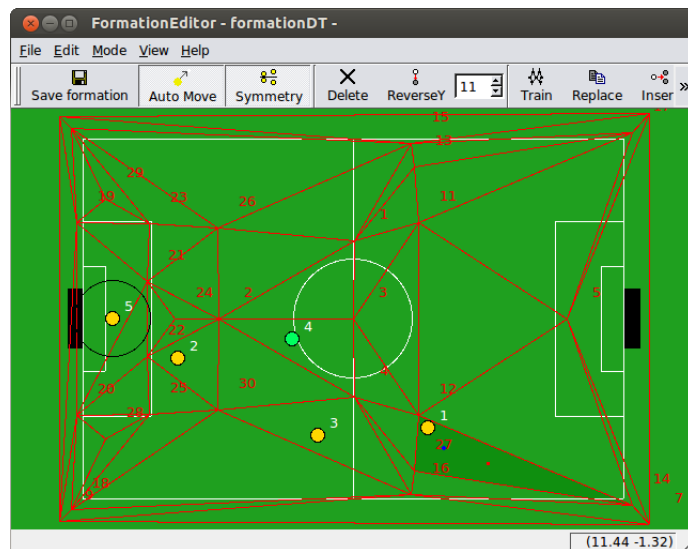


Figura 2.6: Interface de configuração das Formações DT.

Capítulo 3

Trabalho Desenvolvido

Este trabalho consiste na reformulação das tarefas do *Coach*. O novo *Coach* manterá o sincronismo na atribuição de posições estratégicas, que já tinha, com a novidade de decidir qual a melhor formação em cada iteração. Todo o trabalho desenvolvido terá foco no *Coach* já existente e será criada uma nova ferramenta para poder configurar o comportamento do novo *Coach*.

Todo o projeto CAMBADA está desenvolvido em Linux 32 bits sendo que este será o ambiente de desenvolvimento deste projeto, não fazendo assim sentido o uso de qualquer outro ambiente. A linguagem de programação escolhida para o desenvolvimento será C++ para algoritmos e controlo dos programas. A linguagem escolhida foi uma herança deixada pelo antigo *Coach*. Decidiu-se manter esta linguagem por não serem antevistas barreiras ao uso da mesma, sendo possível manter e reutilizar trabalho anterior.

Para o ambiente gráfico foi escolhida a *framework* Qt [9]. Esta é uma *framework* multi-plataforma escrita em C++ criada pela empresa norueguesa Trolltech e mantida atualmente pela empresa finlandesa Digia. Por usar como linguagem base C++ torna-se fácil a integração da *interface* gráfica com os algoritmos do novo *Coach*. Também outras *interfaces* do CAMBADA, nomeadamente a *Base Station*, usam a *framework* Qt.

3.1 CoachConfig

Esta aplicação destina-se a ser usada antes dos confrontos. Permitirá configurar e definir num só ficheiro a configuração seguida pelo *Coach* que irá ser executada em jogo. Cada configuração terá duas listas distintas:

- **Lista de Formações** - Esta lista terá todas formações DT [25] usadas na configuração. Cada formação DT é um ficheiro previamente gravado no sistema de ficheiros. As formações são elaboradas numa interface já existente desenhada para o efeito. O *Coach* apenas terá que manipular as formações enquanto ficheiros individuais.
- **Lista de Regras** - Lista de regras com prioridade. Cada regra terá uma formação associada que será selecionada pelo *Coach* caso os parâmetros da regra avaliada sejam válidos.

3.1.1 Lista de Formações

A lista de formações apenas tem os caminhos para as formações sendo atribuído um nome unívoco a cada formação. O nome, atribuído pelo utilizador, é geralmente explicativo dependendo da natureza ou contexto em que a formação deve ser utilizada. Imagine-se que o utilizador criou uma formação DT com um nome `f_def_4.conf`, atribuindo a esta formação o nome **Defender** tornar-se-á mais perceptível quando se precisar de atribuir uma formação a uma regra.

Esta lista não tem qualquer prioridade. Desta forma as duas únicas funcionalidades desta *interface* são a de adicionar e remover uma formação seleccionada. A Figura 3.1 mostra a lista de formações usada no desafio científico do Robocup 2013.

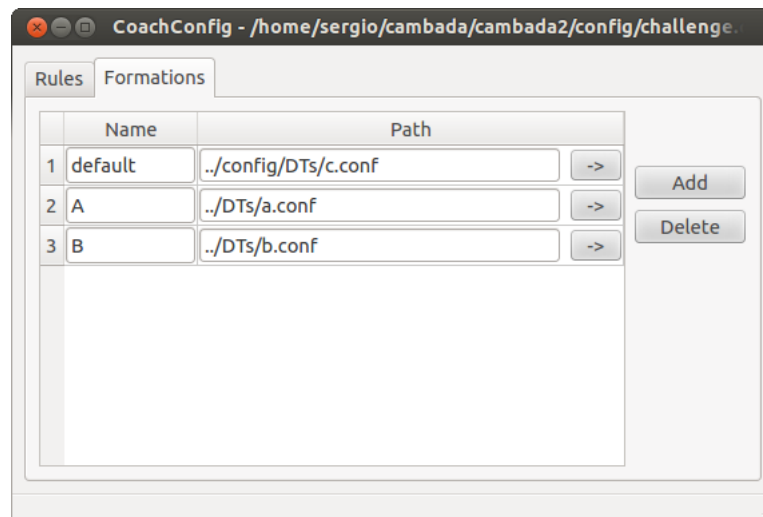


Figura 3.1: CoachConfig - Lista de formações usada no desafio científico do Robocup 2013.

3.1.2 Lista de Regras

Uma regra permite definir uma formação caso determinados parâmetros sejam verdadeiros num dado momento de jogo. O utilizador pode definir tantas regras quantas desejar dependendo da metodologia que preferir utilizar para o jogo. Na *interface* onde se podem parametrizar as regras constam outros parâmetros de cariz global a todas as regras. Estes parâmetros servem de suporte às regras e serão detalhados mais à frente nesta secção. A Figura 3.2 mostra as regras e os parâmetros usados no desafio científico do Robocup 2013.

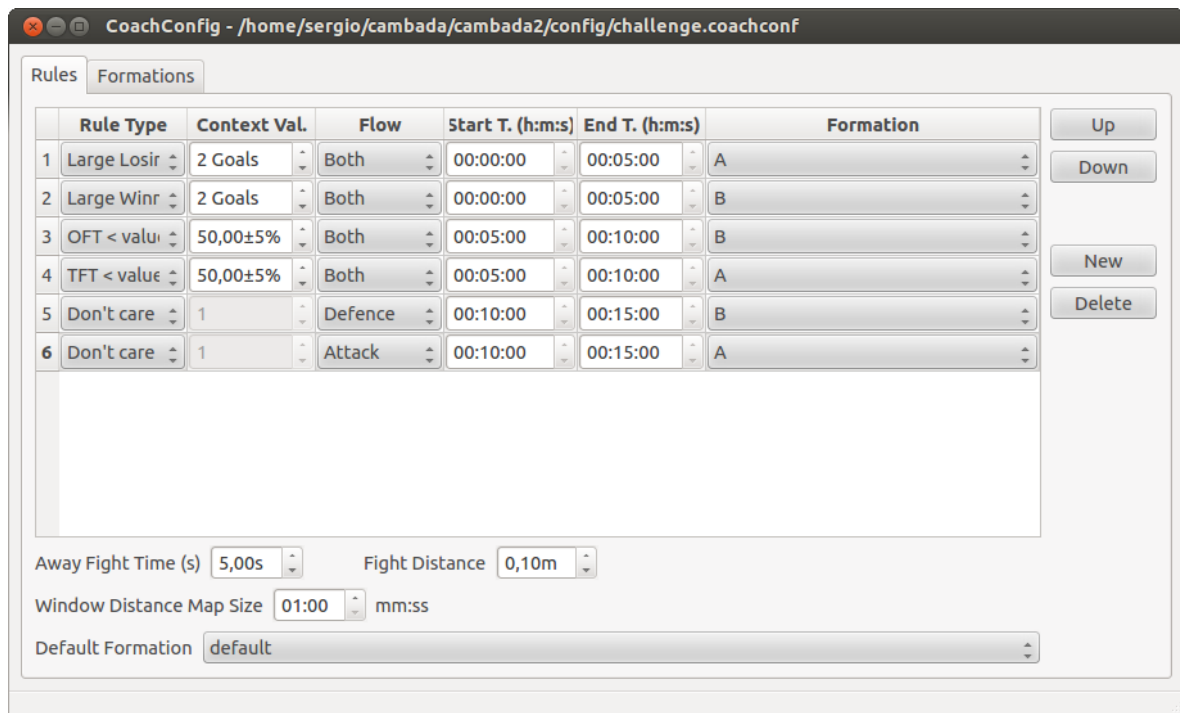


Figura 3.2: CoachConfig - Lista de regras usada no desafio científico do Robocup 2013.

Parâmetros Comuns

Todas as regras têm 4 parâmetros que lhes são transversais:

- **Flow** - Este parâmetro define o estado do jogo para equipa diferenciando se a equipa se encontra a atacar ou a defender. A equipa encontra-se a atacar sempre que tem a bola em sua posse. A equipa encontra-se a defender sempre que a bola estiver demasiado tempo longe de todos os jogadores da mesma. A forma como as transições deste parâmetro ocorrem baseia-se em informações do jogo e está detalhada na secção 3.2.2. Existe a possibilidade deste parâmetro ser ignorado escolhendo *Both*, fazendo com que o parâmetro *Flow* da regra seja verdadeiro tanto no ataque como da defesa.
- **Start Time** e **End Time** - Estes dois parâmetros juntos definem o intervalo de tempo em que a regra é válida. Desta forma é possível definir o instante de tempo em que a regra entra em vigor (*Start Time*) e o instante de tempo em que a regra deixa de fazer sentido (*End Time*). Estes são definidos em **hh:mm:ss** (horas:minutos:segundos) a contar desde o início do confronto.
- **Formation** - Este parâmetro indica a formação previamente configurada na lista de formações pelo nome que lhe foi atribuído. Esta será a formação a ser utilizada caso a regra seja ativada durante o jogo.

Tipos de Regras

As regras configuradas têm para além dos parâmetros acima descritos um parâmetro que varia dependendo do tipo da regra. Este parâmetro é definido na *interface* do *CoachCon-*

fig como *Context Value*. Existem 5 tipos de regras onde este parâmetro assume diferentes significados:

- **Large Losing** - Regra válida quando a equipa se encontra a perder por uma quantidade de golos maior ou igual a um valor definido pelo utilizador.
- **Large Winning** - Regra válida quando a equipa se encontra a ganhar por uma quantidade de golos maior ou igual a um valor definido pelo utilizador.
- **Our Field Time** - Regra válida quando o tempo que bola permanece no campo da equipa em percentagem se situa abaixo dum determinado valor percentual. Este tempo é compreendido numa janela temporal e será detalhado na secção 3.2.2.
- **Their Field Time** - Regra válida quando o tempo que bola permanece no campo do adversário em percentagem se situa abaixo dum determinado valor percentual. Este tempo, da mesma forma que o tipo de regra anterior, é compreendido numa janela temporal e será detalhado na secção 3.2.2.
- **Don't Care** - Regra especial que ignora o campo de *Context Value* fazendo com que a regra apenas obedeça aos parâmetros comuns a todas as regras.

Com a finalidade de simplificar o *interface*, o campo correspondente ao *Context Value* muda a sua formatação consoante o tipo de regra escolhido. A Tabela 3.1 sintetiza os tipos de dados e os intervalos aceites pelo parâmetro *Context Value* em função do tipo de regra.

Tabela 3.1: *Context Value* em função do tipo de regra

Tipo de Regra	Tipo da variável	Intervalo Válido
<i>Don't Care</i>	-	-
<i>Large Losing</i>	Inteiro (Número de Golos)	[1, 20]
<i>Large Winning</i>	Inteiro (Número de Golos)	[1, 20]
<i>Our Field Time</i>	Percentagem de tempo	[0.00%, %100.00]
<i>Their Field Time</i>	Percentagem de tempo	[0.00%, %100.00]

Nas regras *Our Field Time* e *Their Field Time* fixou-se uma percentagem de *threshold* de 5% relativamente ao valor indicado pelo utilizador. Imagine-se que foi definida uma regra *Our Field Time* com um valor de 50%. Sem aplicar um valor de *threshold* a regra estaria ativa sempre que a distribuição se encontra-se no intervalo de 0% a 50%, ficando desativa no resto do intervalo. Com um *threshold* de 5% a regras passará de ativa para inativa se a distribuição ultrapassar o valor de 55%. No sentido oposto a regra apenas passará de inativa para ativa se a distribuição baixar do valor de 45%. Isto evita mudanças demasiado rápidas na formação quando a bola está a ser disputada na zona do meio campo. O gráfico da Figura 3.3 mostra o comportamento duma regra *Our Field Time* em função da distribuição dentro da zona do meio campo da equipa.

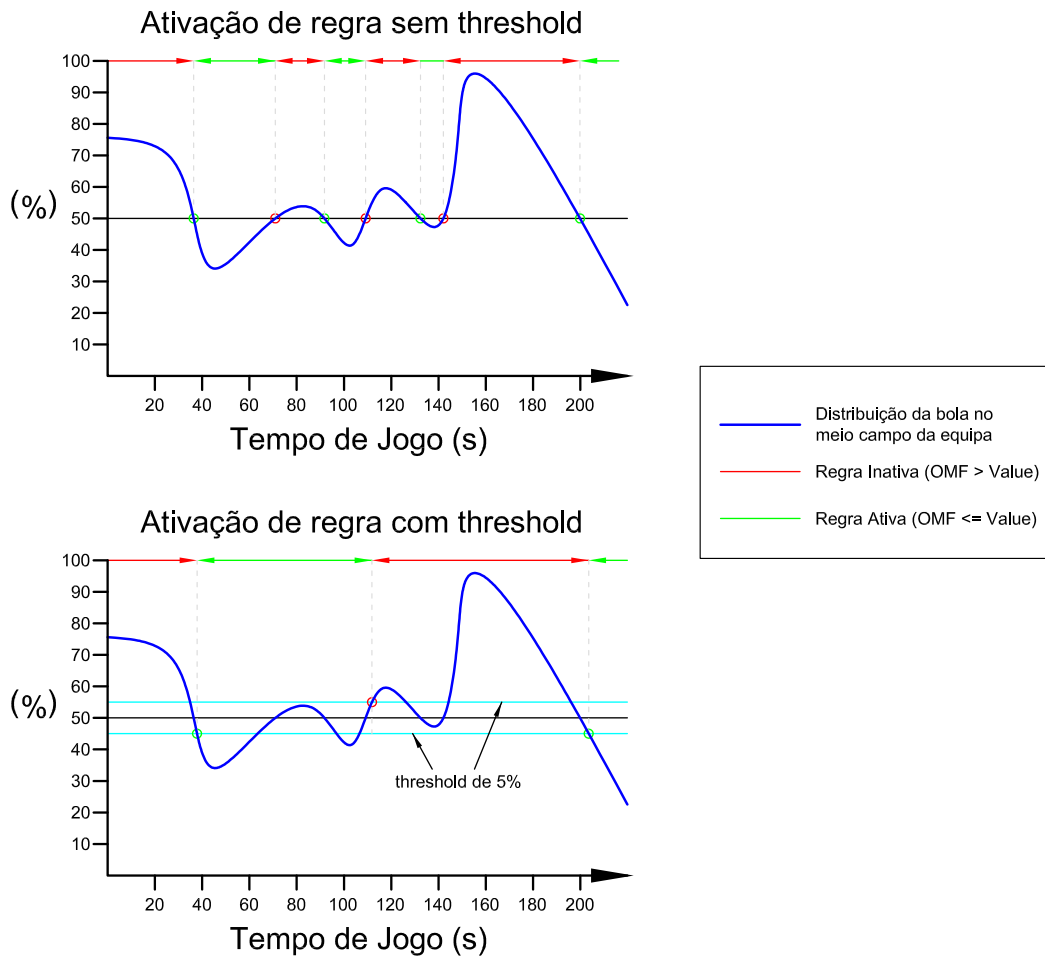


Figura 3.3: Efeito de *threshold* na ativação duma regra *Our Field Time*.

Visto a lista de regras ser uma lista com prioridade, a ordem pela qual são definidas as regras irá alterar o comportamento do *Coach*. Desta forma a *interface* foi dotada dum botão para aumentar a prioridade duma regra (botão *Up*) na lista e outro botão fazer diminuir (botão *Down*). Evita-se desta forma que o utilizador tenha que redefinir as regras sempre que entender mudar a prioridade das regras entre si.

Default Formation

Esta é a formação a ser usada pela equipa nos momentos de jogo em que todas as regras definidas sejam inválidas. A indicação da *Default Formation* é obrigatória e deve indicar uma das formações DT definidas na lista de formações. A existência duma regra por defeito pode facilitar a configuração das regras. O utilizador pode desta forma pensar nas regras como contemplações de situações particulares ao jogo. Por outro lado, o utilizador pode preferir uma metodologia em que cobre todas as situações de jogo com regras diferenciadas. Nesta última metodologia caso exista uma situação que o utilizador não contemplou, esta formação por defeito evita um momento de indefinição de formação a tomar.

3.1.3 Outros parâmetros de configuração

Away Fight Time e Fight Distance

Um novo conceito que foi implementado é o fluxo do jogo (*Flow*). A noção de fluxo de jogo é inferida com base na posição da bola relativamente aos jogadores de campo da equipa. Como já fora indicado na explicação do parâmetro *Flow*, apenas é certo que a equipa se encontra a atacar quando tem a bola na sua posse. É impossível distinguir, pelo menos até momento da criação do novo *Coach*, se a bola está perdida no campo ou se o adversário a tem em sua posse. Foi esta indefinição que determinou a forma como o conceito *Flow* foi implementado e que por sua vez deu origem aos parâmetros *Away Fight Time* e *Fight Distance*.

Antes da implementação do conceito de *Flow* foram detetados à partida dois aspetos importantes que tiveram que ser tomados em conta:

- **Posse de bola instável**

- **Problema:** Os dispositivos que são usados pelos agentes para segurar a bola em sua posse (*Grabbers*) não são bem comportados. Durante o drible da bola os *Grabbers* nem sempre seguram a bola na perfeição criando uma intermitência relativamente à posse de bola. Tal intermitência não transmite o conceito que queremos extrair, pois para o robô o drible é um processo bivalente entre segurar a bola e deslocá-la para um determinado ponto. Também quando um robô está a disputar a bola com um robô adversário a posse de bola não é bem definida. Caso estas situações não fossem previstas ocorreriam muitas trocas de fluxo indesejadas.
- **Solução:** Para contrariar o problema foi proposto um parâmetro de configuração denominado *Fight Distance*. Este consiste num valor real que representa a distância máxima em metros que o robô deve considerar que a bola ainda se encontra na sua posse. Uma vez a atacar a equipa apenas passa a defender caso a bola se afaste mais do que a *Fight Distance* relativamente a qualquer robô em jogo.

- **Perca da bola intencional**

- **Problema:** Sempre que é executado um passe entre dois robôs a bola não só deixa de estar na posse dos dois robôs como também pode ficar muito distante dos mesmos. Se esta situação não fosse prevista sempre que a equipa executasse um passe o fluxo iria passar pelo estado de defesa durante a viagem da bola. Tal acontecimento iria quebrar o posicionamento dos robôs no ataque dificultando assim o avanço no campo.
- **Solução:** Para contrariar o problema foi proposto um parâmetro de configuração denominado *Away Fight Time*. Este consiste num intervalo de tempo em que deve ser mantido o fluxo de ataque mesmo que a bola se afaste muito dos robôs. Permite-se assim que a bola viaje da zona de *Fight Distance* do robô que efetua o passe até à zona de *Fight Distance* do robô que vai receber a bola.

Na secção 3.2 será explicado como são usados estes dois parâmetros.

Window Distance Map Size

Pretende-se obter através dum valor percentual a quantidade de tempo que a bola se encontra no meio campo da equipa. Caso se quisesse manter toda a informação desde o início

do jogo, a posição da bola de momentos mais recentes dificilmente iria fazer-se sentir em momentos mais tardios, tornando-se pouco útil contabilizar a posição da bola ao longo do jogo.

O parâmetro *Window Distance Map Size* define um intervalo de tempo máximo, definindo o momento em que o jogo se encontra e o momento mais antigo em que a posição da bola deve ser monitorizada. Na secção 3.2 será explicado como foi implementado este mecanismo. Fica ao critério do utilizador definir qual o melhor valor para este parâmetro podendo variar desde um segundo até 30 minutos (2 vezes o tempo dum parte num jogo da MSL).

3.1.4 Menus e Operações

O *CoachConfig* tem um menu onde podem ser escolhidas as seguintes operações:

- **New configuration** (Nova configuração) - Descarta as configurações atuais e carrega uma nova configuração.
- **Open** (Abrir) - Abre uma configuração criada anteriormente.
- **Save** (Guardar) - Guarda a configuração atual.
- **Save as** (Guardar como) - Guarda a configuração atual num ficheiro indicado pelo utilizador.
- **Configure Coach** (Configurar o *Coach*) - Executa todas as ações necessárias para configurar o *Coach* com os parâmetros atuais do *CoachConfig*. Até este ponto nenhuma parametrização fora refletida no novo *Coach*. É depois desta ação que o *Coach* passa a estar parametrizado. A secção 3.1.4 irá aprofundar mais esta ação.
- **Close** (Sair) - Fecha o *CoachConfig*.

O ficheiro de persistência usado para guardar as configurações realizadas no *CoachConfig* é um ficheiro XML com extensão `*.coachconfig`. Sempre que se tentar guardar uma configuração a aplicação verifica algumas premissas de consistência:

- Existe de pelo menos 1 formação DT na lista das formações.
- *Default Formation* está definida.
- Todas as regras criadas têm uma formação definida.

Se alguma destas verificações falhar o utilizador é notificado devidamente tendo que regularizar a sua configuração. De seguida terá de voltar a guardar a configuração onde se repete o processo de verificação das premissas.

Interoperabilidade *CoachConfig/Coach*

Para garantir a interoperabilidade entre as aplicações usou-se uma árvore de objetos definidos num ficheiro *XML Schema Definition* (XSD). Este, depois de definido na conceção dos parâmetros passados entre os programas, serviu para gerar as classes de C++ usadas para ler e escrever ficheiros `*.coachconfig`.

Para criar as classes de manipulação de XML foi usada a plataforma "CodeSynthesis XSD" [4]. Esta plataforma gera de forma automática as classes C++ que permitem ler

ficheiros XML para memória e vice-versa. O ficheiro XSD do Apêndice A foi o usado para criar as classe de manipulação dos ficheiros `*.coachconfig`.

Com este ficheiro foram gerados dois ficheiros: `CoachConfigFile.cxx` e `CoachConfigFile.hxx` que implementam os objetos de manipulação de XML e as suas *interfaces* respetivamente. O Apêndice B serve como exemplo de configuração do *Coach*, tendo sido usado no desafio científico do Robocup 2013.

Configure Coach

O CAMBADA usa a mesma estrutura de ficheiros em todos os computadores: computadores de desenvolvimento, computador da *Base Station* e computadores dos robôs de campo. Desta forma, e com intuito de simplificar o uso das configurações, foram centralizados todos os ficheiros resultantes na pasta `~/config/FormationStore`. A Figura 3.4 mostra a proveniência dos ficheiros, quem os utiliza e de que forma.

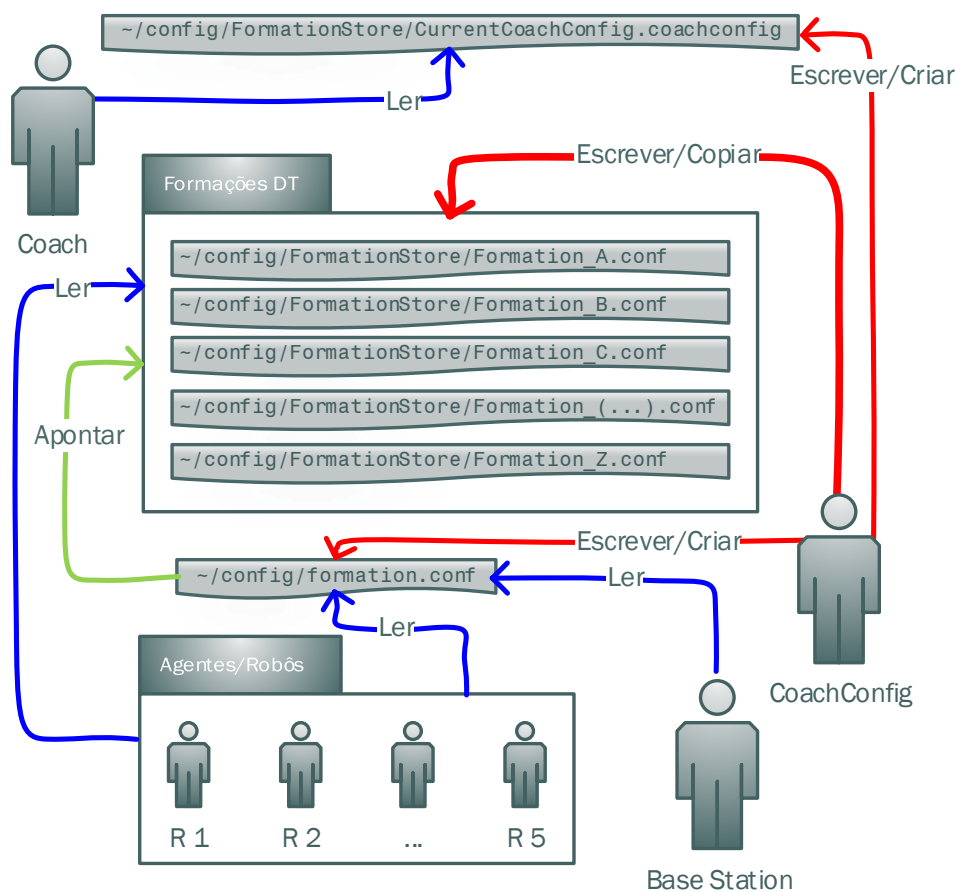


Figura 3.4: Treinador automático no CAMBADA - Leitura e escrita de ficheiros.

O único ficheiro que não se encontra na pasta `~/config/FormationStore`, estando na

pasta acima da árvore, é o ficheiro `formations.conf`. Este já era utilizado anteriormente pelos agentes para indicar qual a formação DT a ser utilizada de forma estática antes do novo *Coach*. A estrutura deste ficheiro de texto é muito simples: a primeira linha tem um número inteiro que indica quantas formações estão definidas e cada uma das linhas que se seguem tem dois literais onde o primeiro indica o nome dado à formação e o segundo o caminho do ficheiro onde está a definida a formação DT. A *Base Station* lê o ficheiro `formations.conf` para saber os nomes das formações em uso pois irá ter algum controlo no processo como será explicado com maior detalhe na secção 3.3. Os agentes leem o ficheiro `formations.conf` para saberem os caminhos das formações DT que irão usar, desta forma será apenas indicado o ID da formação a usar num determinado momento de jogo via RTDB.

Quando se selecciona a opção `Configure Coach` no *CoachConfig* este irá eliminar o conteúdo da pasta `~/config/FormationStore`. Uma vez a pasta vazia é criado com base na configuração atual o ficheiro `CurrentCoachConfig.coachconf` copiando também para o mesmo local todas formações DT envolvidas. Por último reescreve o ficheiro `formations.conf` por forma a ficar consistente com a nova configuração. O ficheiro `formations.conf` passa a apontar para as formações DT acabadas de copiar e não para as suas localizações originais. Estes ficheiros apenas são escritos na altura em que se selecciona esta opção do *CoachConfig* sendo que durante o jogo serão apenas acedidos para leitura.

3.2 Novo *Coach*

O novo *Coach* foi pensado para não necessitar de intervenção humana durante o decorrer do jogo. Quando é lançado carrega o ficheiro `CurrentCoachConfig.coachconf` onde aprende todas as suas definições ficando imediatamente ativo.

A *interface* gráfica está dividida em duas colunas. Na primeira coluna podemos escolher se queremos ver a distribuição da bola no campo de forma gráfica ou se queremos ver uma tabela dinâmica que mostra a avaliação das regras em vigor. As Figuras 3.5 e 3.6 mostram a interface do novo *Coach*, onde a primeira mostra a distribuição da bola no campo e a segunda mostra a tabela dinâmica da avaliação das regras.

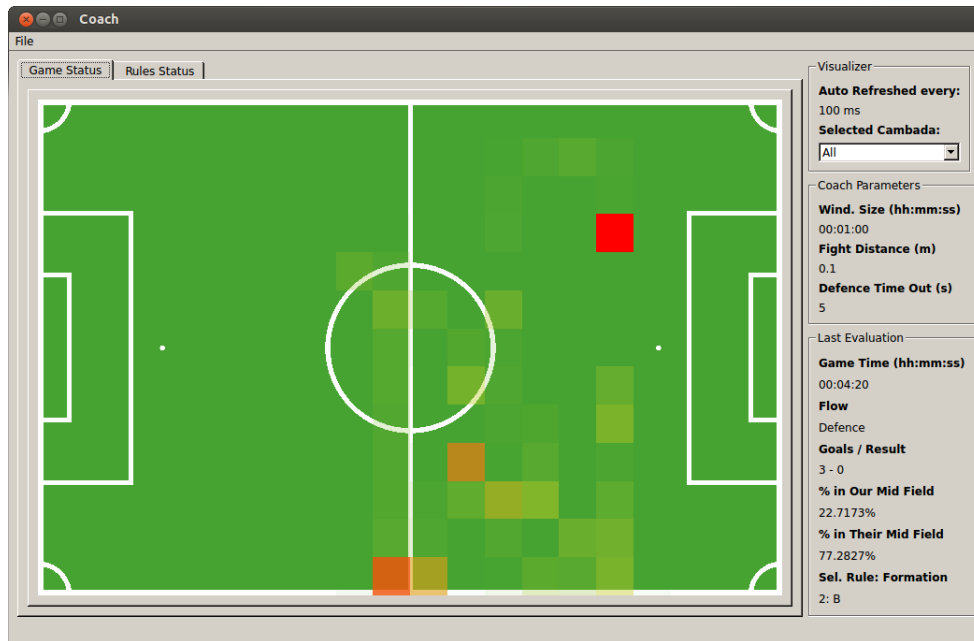


Figura 3.5: Coach - Interface de amostragem de distribuição da bola pelo campo.

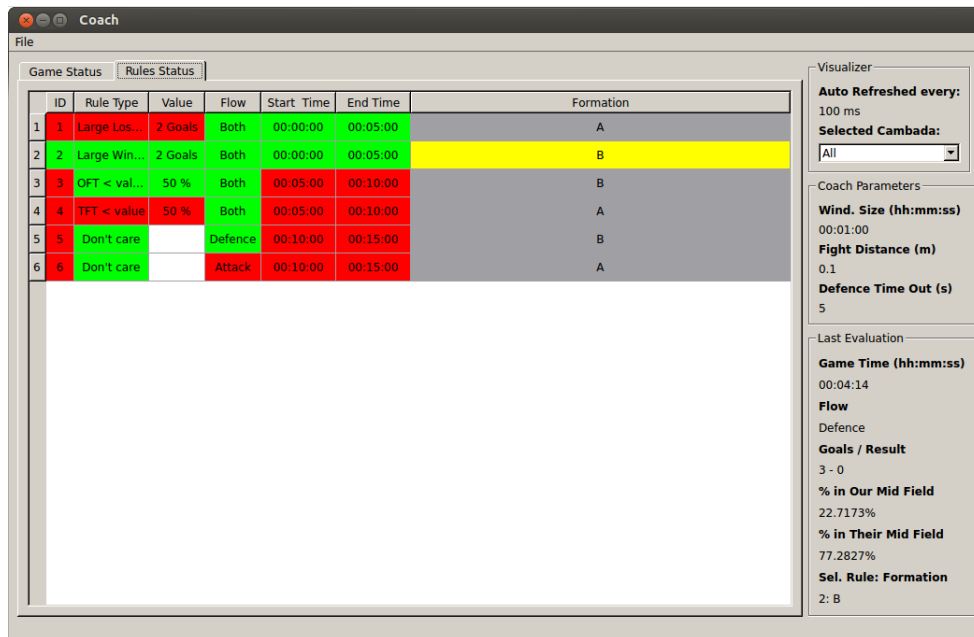


Figura 3.6: Coach - Tabela de amostragem do resultado da avaliação das regras usadas no desafio científico do Robocup 2013.

Durante o jogo irá estar visível a tabela dinâmica pois é onde todas as regras estarão a ser avaliadas. Na segunda coluna os campos apresentados são fixos e dividem-se em três grupos:

- **Visualização** - Aqui escolhemos se queremos ver a distribuição de jogo anunciada por todos os jogadores ou se apenas por um jogador (*Selected Cambada*). Durante

o jogo interessa ver apenas a média de todos os jogadores, pois é este valor que é levado em conta durante as avaliações das regras. Também nos é dada a velocidade de refrescamento do mapa de distribuição de bola no campo (*Auto refreshed every*).

- **Parâmetros do Coach** - Estes parâmetros são fixos durante o jogo e foram definidos na configuração atual (*Window Size, Fight Distance e Defence Time Out*).
- **Dados da última avaliação** - Neste grupo são mostrados os dados de *input*, *output* e os dados estatísticos obtidos desde a última avaliação do *Coach*. O tempo de jogo (*Game Time*) é o um dado de *input* anunciado pela *Base Station* assim como o resultado do jogo (*Goals/Result*). Os restantes parâmetros de *input* (*flow* e *% of time in field*) para a avaliação das regras são obtidos via estatística e serão explicados com maior detalhe mais à frente neste capítulo. Como *output* é nos indicado qual a formação atual em vigor (*Sel. Rule: Formation*).

Na imagem 3.5 podemos observar que por cima do desenho do campo está sobreposta uma grelha de quadrados vermelhos. Esta grelha tem uma dimensão de 20 quadrados largura por 13 quadrados de altura. Cada quadrado representa a zona por ele delimitado sendo que o quadro totalmente vermelho indica a zona onde a bola esteve mais tempo. Quanto menos tempo estiver a bola numa zona maior será a transparência da coloração vermelha do quadrado que delimita essa mesma zona. Na tabela dinâmica teremos a amostragem do resultado da avaliação das regras. Cada parâmetro será colorido seguindo um esquema de cores a fim de se constatar o resultado da última avaliação. Sempre que um parâmetro for avaliado e for válido será representado com uma cor verde. Caso seja inválido será representado com uma cor vermelha. Na última coluna desta tabela a regra que se encontrar ativa terá a sua formação colorida com fundo amarelo tendo as outras regras um fundo cinzento. Caso nenhuma das regras se encontre ativa todas as formações das regras serão coloridas a cinzento.

3.2.1 Funcionamento

Depois de todos os parâmetros configurados o *Coach* entra num ciclo infinito até que seja interrompido. O ciclo compreende uma iteração completa das funções do *Coach* sendo executado à frequência de aproximadamente 30 Hz, controlado com uma função de *sleep* de 33 ms. Este valor foi ponderado com base na frequência de atualização da RTDB, pois não faria sentido produzir dados estatísticos a uma taxa superior à taxa de refrescamento da RTDB.

O ciclo começa por atualizar todos os dados de *input*. De seguida avalia todas as regras até que uma das regras seja avaliada positivamente ou até que as regras se esgotem. Após ser encontrada uma regra em que todos os parâmetros são válidos é utilizada a formação DT da regra. No caso em que a situação anterior não acontece, ficando sem regras para avaliar, é escolhida a formação por defeito (*Default Formation*). Antes do *Coach* enviar o seu resultado para a equipa verifica se o modo de escolha da formação atual se encontra em modo automático ou em modo manual, caso o último caso se verifique o *Coach* não anuncia a formação DT na RTDB. Isto acontece porque a *Base Station* foi alterada para este projeto de modo a permitir inibir o sistema implementado. É então possível definir a formação manualmente ignorando os resultados provenientes do *Coach*. Este mecanismo será tratado mais à frente na secção 3.3. A Figura 3.7 sintetiza um ciclo desta iteração.

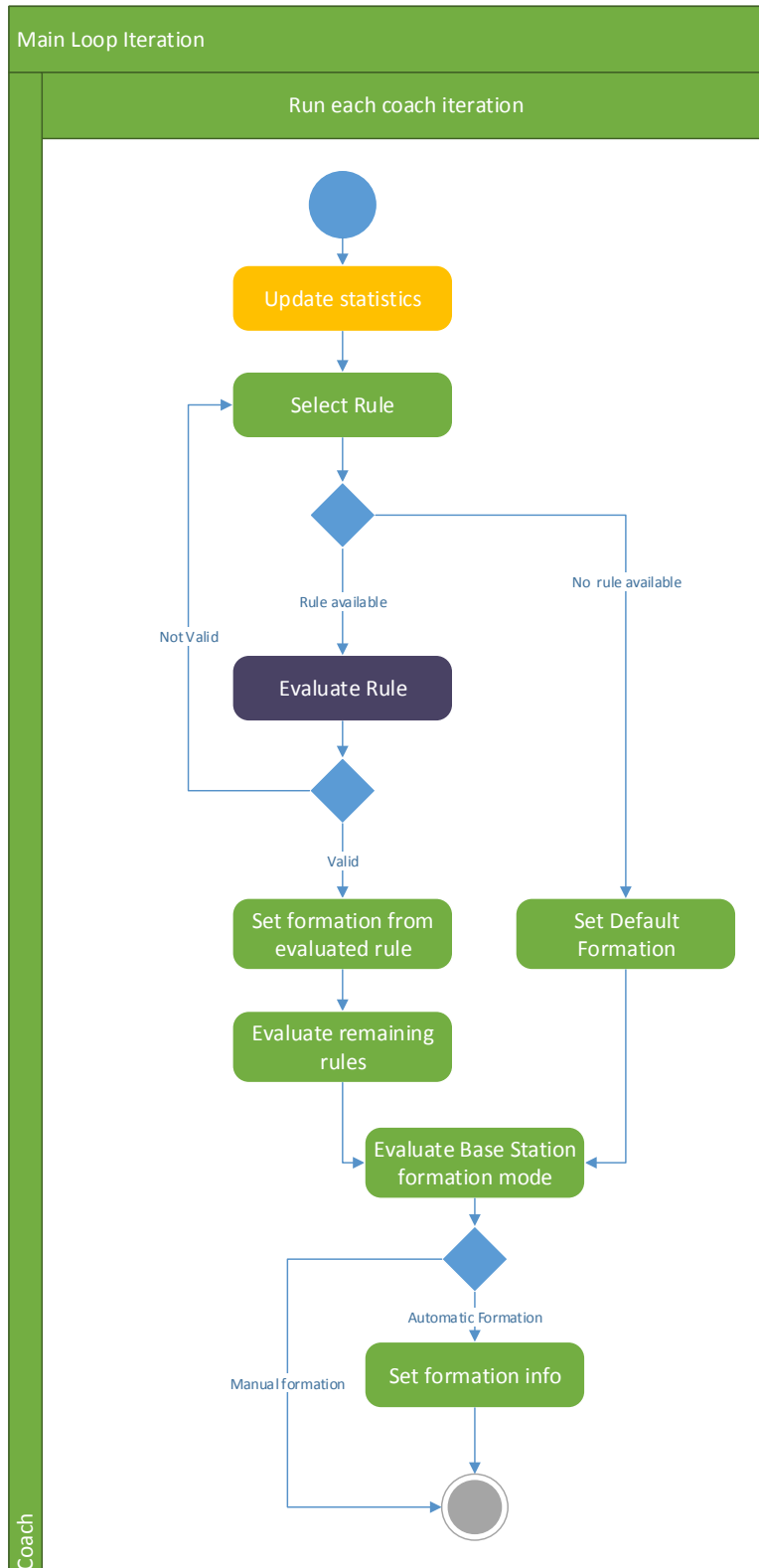


Figura 3.7: Coach - Fluxo da iteração principal.

3.2.2 Obtenção do dados de entrada

A primeira tarefa de cada iteração do *Coach* é a atualização dos dados de *input* assim como manutenção dos dados estatísticos. Todos os dados são obtidos diretamente da RTDB à exceção do estado de *Flow* e dos dados estatísticos relativos à distribuição da bola. A Figura 3.8 mostra o fluxo de execução desta função.

Flow

Como já tinha sido indicado anteriormente este é um novo parâmetro utilizado para indicar à equipa se está em situação atacante ou defensora. Visto que os robôs não têm a capacidade de identificar com precisão se a bola está na posse do adversário ou se está perdida no campo a lógica deste conceito teve de ser adaptada às limitações técnicas atuais do CAMBADA.

Foi criada uma máquina de estados denominada *Flow State Machine* (FSM) com três estados:

1. A atacar (*Attacking*):

Apenas ocorre uma transição para este estado quando bola for apanhada por um elemento da equipa. Enquanto a distância mínima de todos os agentes à bola for inferior ao parâmetro *Fighting Distance* este estado é mantido, caso contrario existe uma transição para o estado *Counting Down*, dando-se neste instante início de contagem dum *Timer* decrescente. A quantidade de tempo deste *Timer* é definida no parâmetro de configuração *Away Fight Time*.

O *output* deste estado é *Attack*.

2. Em contagem decrescente (*Counting Down*):

Uma vez neste estado, o *Coach* verifica a cada iteração o estado do *Timer* previamente iniciado na saída do estado anterior. Caso o tempo restante do *Timer* termine ocorrerá uma transição para o estado *Defending*.

O *output* deste estado é *Attack* a fim de permitir passes quando a equipa se encontra a atacar.

3. A defender (*Defending*):

Nesta situação apenas se espera que a equipa recupere a bola para se poder passar para o ataque.

O *output* deste estado é *Defence*.

Na Figura 3.8 podemos ver as transições da FSM acima descritas.

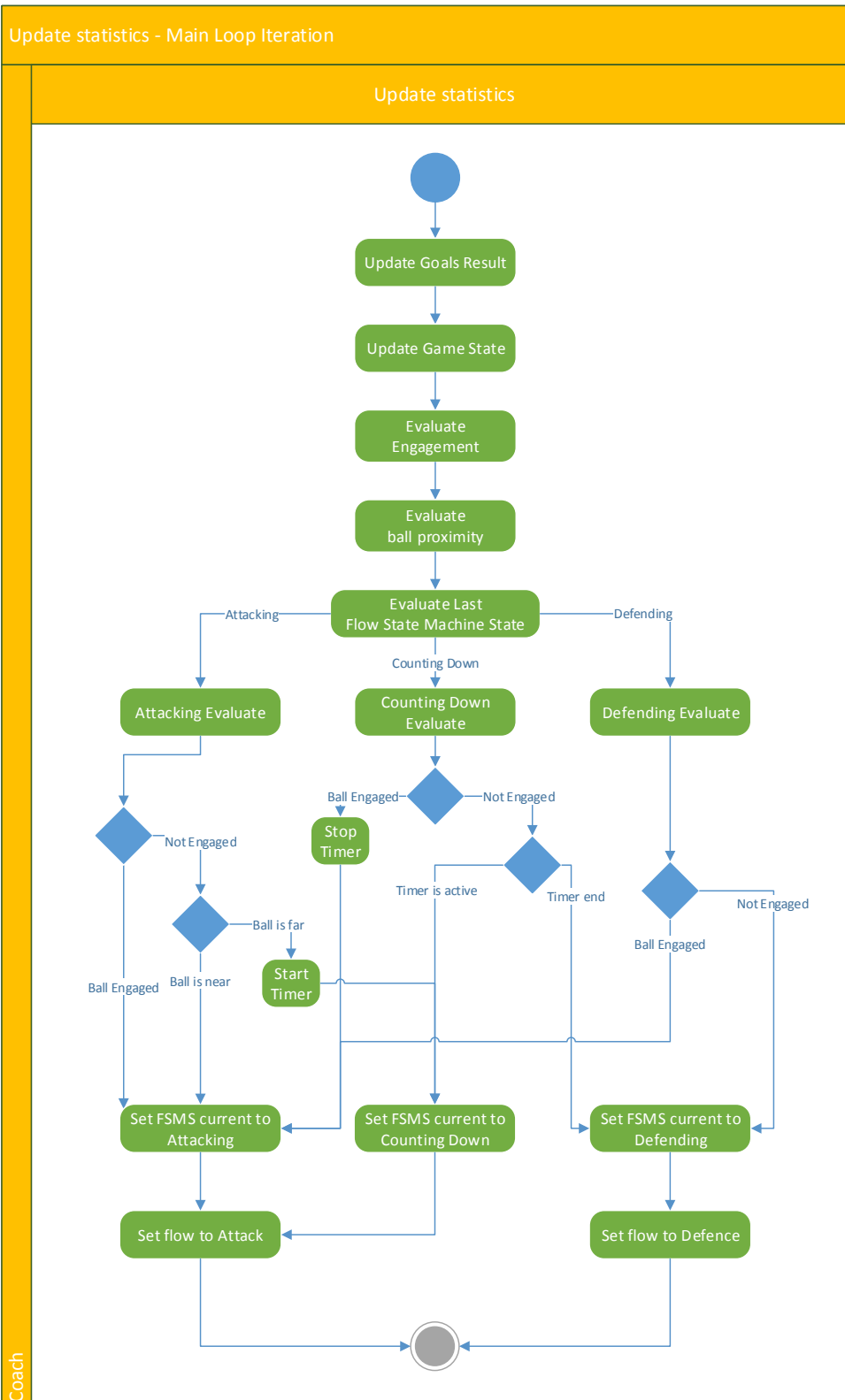


Figura 3.8: Coach - Fluxo da função UpdateStatistics.

Mapa de distribuição da bola no campo

A obtenção de dados estatísticos sobre localização dum objeto num plano a duas dimensões passa, duma forma geral, pelo registo das posições discretas nesse mesmo plano e análise das mesmas. Uma das estratégias mais comuns consiste em registar as posições obtidas ao longo do tempo numa fila. Este método torna-se bastante útil quando queremos fazer um registo com o número limitado de valores descartando os valores mais antigos em detrimento dos valores mais recentes. Porém, quando se pretende obter dados estatísticos torna-se bastante penoso percorrer todos os valores da fila. Uma forma mais eficaz de obter resultados será o registo das posições sobre a forma de matriz. Porém esta técnica tem a desvantagem de não ser natural o registo temporal dos dados introduzidos pois, para isso é necessário manter na matriz não só o peso das posições como também todos os tempos de entrada.

Visto o ciclo principal do *Coach* ser controlado por uma função de *sleep* o processamento de dados estatísticos com matrizes duma forma exaustiva poderia revelar-se muito pesado. É também desejável que a reserva de memória no *Coach* em *run time* seja minimizada ao máximo para que o fluxo seja bem comportado em termos de tempo de execução. Pretende-se também evitar ao máximo operações em massa com valores em vírgula flutuante visto as operações com números inteiros ser mais eficiente em processadores de uso geral.

Tendo em conta as considerações acima descritas desenhou-se uma solução que visa ser eficiente em termos de gestão de memória e eficiente na hora de obtenção de dados estatísticos. A solução proposta foi implementada recorrendo a um objeto denominado *DistMap*. Este tem uma matriz de inteiros de resolução fixa de 20x13 que inicia com todos os seus valores a zero. Sempre que lhe é pedido para armazenar uma posição no plano este incrementa o valor da matriz referente à posição indicada mantendo separadamente o número total de entradas que existem na matriz. Existe um limiar para o número máximo de valores que a matriz deve ter no total. Esse valor é obtido através da fórmula:

$$Counter_{max} = WindowSize_{(s)} \times RefreshFrequency_{(Hz)} \quad (3.1)$$

O parâmetro *WindowSize* está definido na configuração no parâmetro *WindowTimeInSec*. A frequência esta é fixa em 30 Hz pelas razões já descritas na secção 3.2.1.

Sempre que o número de posições anotadas atinge o valor máximo permitido todos os valores da matriz maiores que zero são decrementados em uma unidade. O contador é assim diminuído em tantas unidades quantas sejam as posições na matriz cujo valor da posição seja diferente de zero. Os valores seguintes a chegar serão introduzidos normalmente repetindo este ciclo. Desta forma são beneficiadas as posições onde a incidência for maior num curto espaço de tempo.

Quando se requer os dados estatísticos referentes a uma determinada zona apenas é necessário somar os valores das regiões pertencentes e dividir pelo número total de valores armazenados na matriz.

3.2.3 Avaliação das regras e propagação dos resultados

A classe *CoachRule* implementa a lógica das regras do *Coach*. É instanciada uma classe por cada regra da definição quando o *Coach* é lançado, sendo todas as instâncias colocadas numa lista por ordem crescente de ID. A classe *CoachRule* implementa um método *validate* que

tem como parâmetros de entrada todos os valores obtidos na função `CoachUpdateStatistics` (resultado do jogo, *Flow*, distribuição da bola, tempo de jogo) e retorna um valor lógico que indica se a regra tem todas as suas condições válidas. As definições das regras são testadas por ordem e basta falhar um dos testes para que os restantes não sejam testados. Desta forma minimizamos os testes efetuados nesta fase. A Figura 3.9 mostra precisamente o fluxo de teste do método `validate` da classe `CoachRule`.

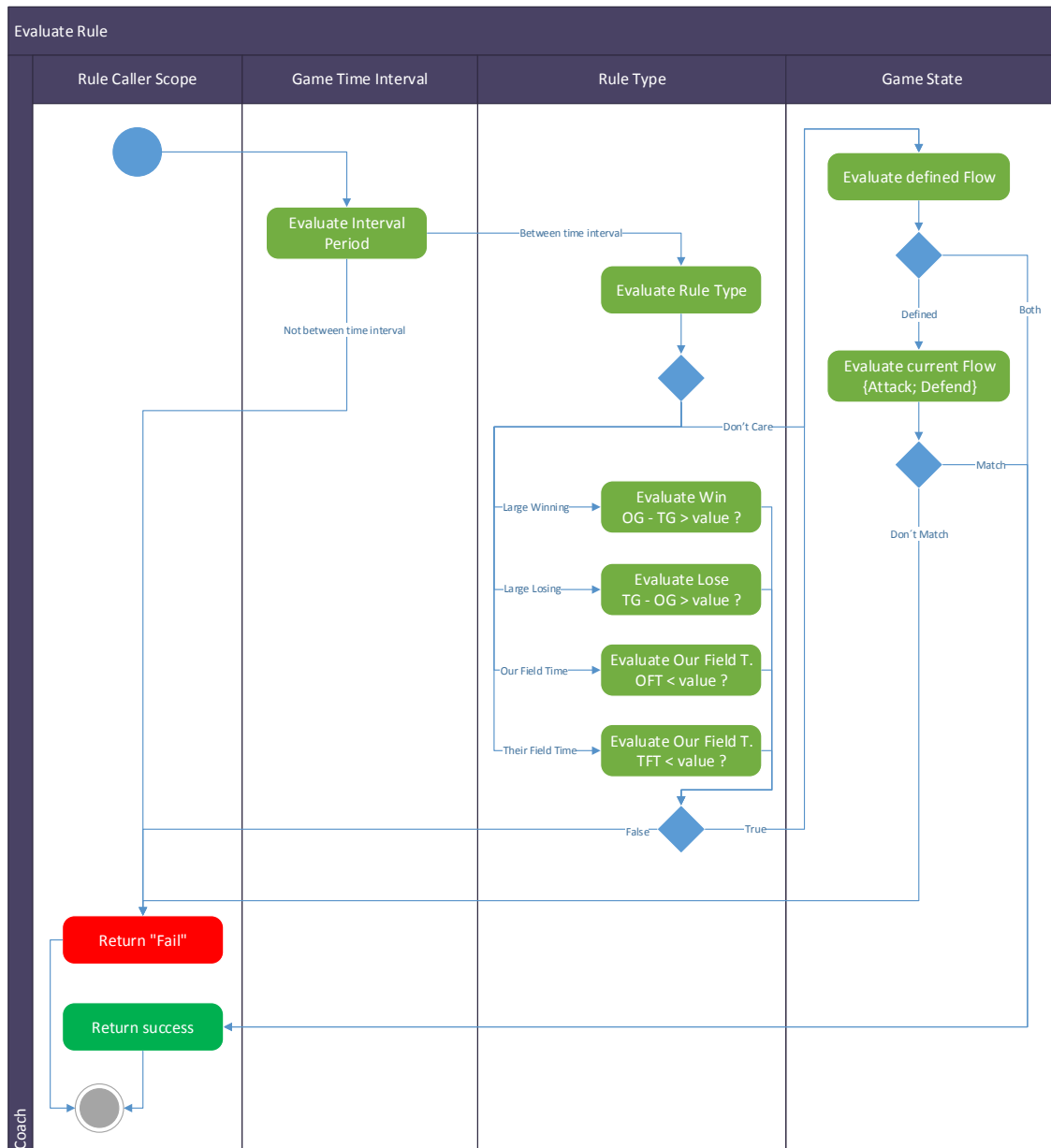


Figura 3.9: Coach - Fluxo da função Evaluate Rule.

O *Coach* comunica com os restantes intervenientes através da RTDB. A informação que transmite é apenas o ID da formação DT que apura em cada iteração e está definido na

estrutura `FormationInfo` no campo `formationID` da RTDB.

3.3 Alterações à *Base Station*

Sempre que o *Coach* altera a formação DT em vigor esta pode ser visualizada na *Base Station*. Para isso foi adicionada uma *Combo Box* (Figura 3.10) que muda de acordo com a formação atual.

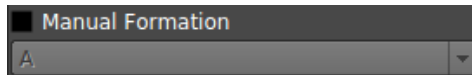


Figura 3.10: Base Station - Modo de formação automático permitindo visualizar qual a formação DT em vigor.

Para manter a lógica de controlo da *Base Station* foi adicionada a hipótese de selecionar manualmente qual a formação a usar (Figura 3.11) ignorando os resultados do provenientes do *Coach*.

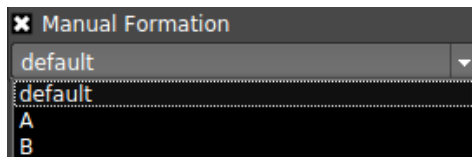


Figura 3.11: Base Station - Modo de formação manual permitindo selecionar qual a formação DT em vigor.

Para que o *Coach* não altere o valor do `formationID` imposto pela *Base Station* em modo de formação manual o modo selecionado é indicado na RTDB. Esta informação é imposta pela *Base Station* na classe `CoachInfo` no campo `manualFormation`. Sempre que este valor está ativo na RTDB o *Coach* não escreve na RTDB deixando a atualização do campo `formationID` a cargo da *Base Station*.

Capítulo 4

Resultados Experimentais

Os testes efetuados ao sistema desenvolvido foram incrementais. Inicialmente testou-se até que ponto os agentes se iriam comportar às mudanças bruscas da formação DT em vigor. Caso este teste falhasse a ordem de trabalhos teria que ser alterada a fim de permitir tal acontecimento. Foram criadas duas formações completamente distintas para ser fácil verificar visualmente que os agentes estavam a mudar a sua formação de jogo. O *Coach* nesta fase apenas trocava entre estas duas formações de 1 em 1 minuto.

Depois de verificado no simulador do CAMBADA, onde as trocas de formações eram evidentes, testou-se este comportamento em campo durante o evento Robótica 2013 em Lisboa. Os agentes comportarão-se como o previsto, sendo as mudanças de formação perfeitamente visíveis e sem atrasos perceptíveis. Partiu-se então para a definição e concretização das regras possíveis de se configurar.

As equipas de futebol robótico na liga MSL têm por experiência que os verdadeiros problemas surgem quando se defrontam contra equipas adversárias e não quando estão a trabalhar nos seus laboratórios. A formação dos jogadores em campo é um fator vital para qualquer jogo em equipa. Assim sendo, optou-se que o trabalho passasse por uma fase simplista mas robusta. A forma como foram pensadas as regras inicialmente contemplaram a importância de se poder testar todas as situações que elas oferecessem.

No RoboCup 2013 o CAMBADA contou com o funcionamento do sistema em pleno. Para cada confronto foram criadas configurações diferentes com duas formações diferentes.

4.1 Desafio Científico

A liga MSL do RoboCup 2013 contou com 2 desafios para além dos jogos entre equipas. Um desses mesmos desafios é o Desafio Científico. Neste desafio as equipas são convidadas a mostrar as suas linhas de investigação mais recentes no âmbito do futebol robótico.

A pontuação do trabalho mostrado por cada equipa é avaliado de 1 a 10 pelas restantes equipas com a supervisão do comité técnico do RoboCup, onde os critérios avaliados serão:

- Qualidade da apresentação
- Interesse tanto para o presente como para o futuro da liga
- Complexidade científica e tecnológica
- Relevância científica para a MSL

- Importância dos resultados experimentais demonstrados
- Relevância dos resultados publicados como suporte para o desafio

Após os votos das equipas os resultados são obtidos através dos seguintes cálculos:

1. É calculada a média das pontuações atribuídas por cada um dos líderes das equipas ($TLav$).
2. É calculada a média das pontuações atribuídas a todas as equipas no desafio:

$$CLav = \frac{\sum_{i=1}^n TLav_i}{n}$$

3. Cada uma das classificações obtidas pelas equipas serão multiplicadas pelo seguinte fator:

$$Cr_i = \frac{CLav}{TLav_i}$$

4. O resultado final de cada equipa será a soma de todos os seus Cr_i

A equipa CAMBADA decidiu levar o seu novo *Coach* para o seu Desafio Científico do RoboCup 2013. Foi sem dúvida uma boa oportunidade para poder submeter este trabalho à crítica das principais equipas de futebol robótico a nível mundial. A prova tem uma duração de 15 minutos onde cada equipa gere o tempo da forma que achar mais adequado de acordo com o que pretende demonstrar.

4.1.1 Apresentação do Desafio Científico

A apresentação do novo *Coach* contou com um conjunto de slides e uma demonstração do sistema em funcionamento.

O conjunto de slides, presentes no Apêndice E, introduziu os elementos que precisam de se sincronizar num desafio da MSL (papeis, controlo da formação e sequência de ações). De seguida foram apresentados os objetivos a que se propõe o novo *Coach*, sendo referida a capacidade de indicar a formação DT a usar dependendo dum conjunto pré-estabelecido de regras.

Para a demonstração do sistema, que teve lugar nos últimos 10 minutos da prova, foram usadas duas formações para além da formação por defeito (Figura 4.3). Estas formações foram pensadas para criar duas atitudes de posicionamento no campo distintas, sendo a formação A (Figura 4.1) uma formação de ataque e a formação B (Figura 4.2) uma formação de defesa. A lista das regras foi preparada a fim de mostrar ativos todos os tipos de regras que é possível usar. Em 3 intervalos de 5 minutos cada mostraram-se todos os tipos de regras possíveis, onde em cada intervalo de tempo eram avaliadas duas regras diferentes. No primeiro intervalo foram mostrados os tipos *Large Losing* e *Large Winning*, no segundo intervalo os tipos *Our Field Time* e *Their Field Time* e por último foram mostradas duas regras *Don't Care* mudando o apenas o *Flow* entre o estado *Attack* e o estado *Defense*.

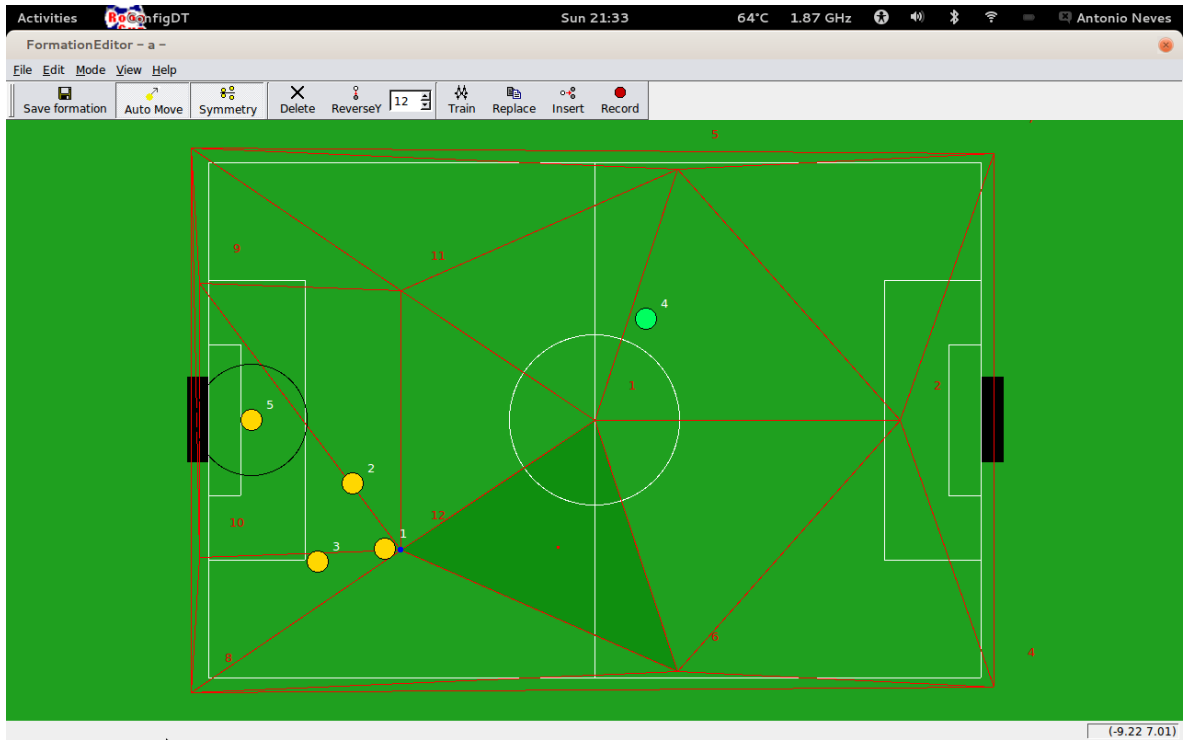


Figura 4.1: Desafio Científico - Formação A

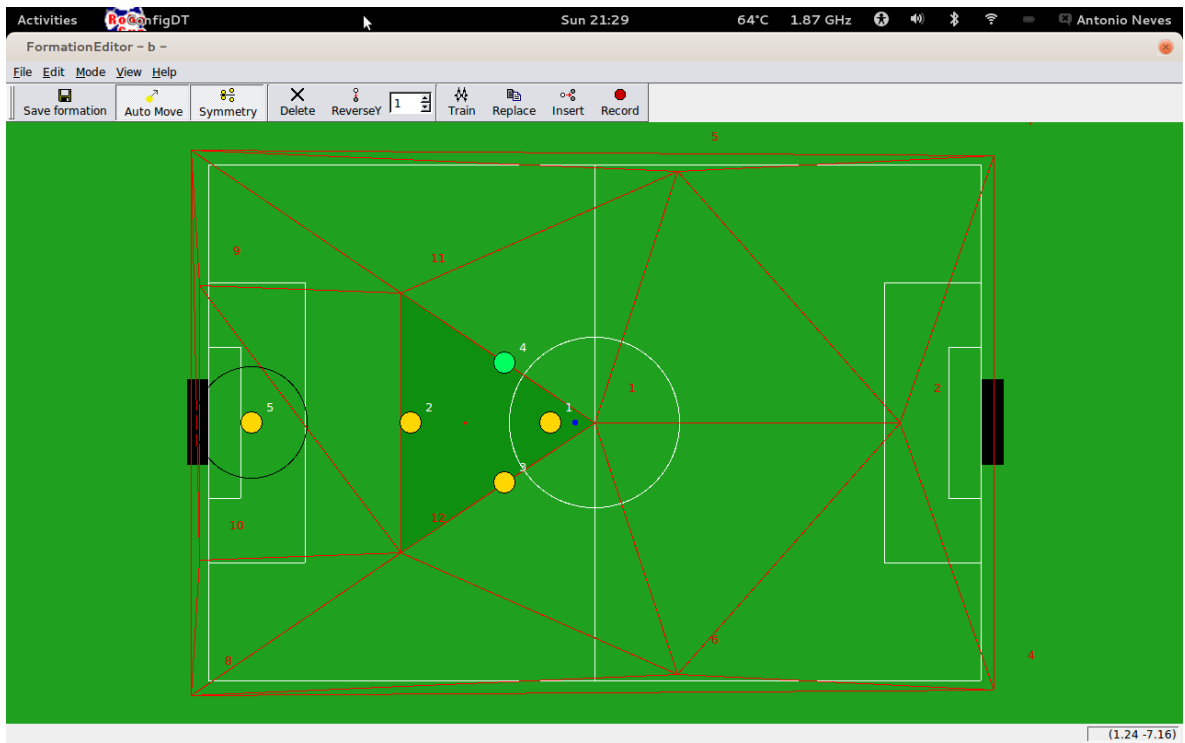


Figura 4.2: Desafio Científico - Formação B

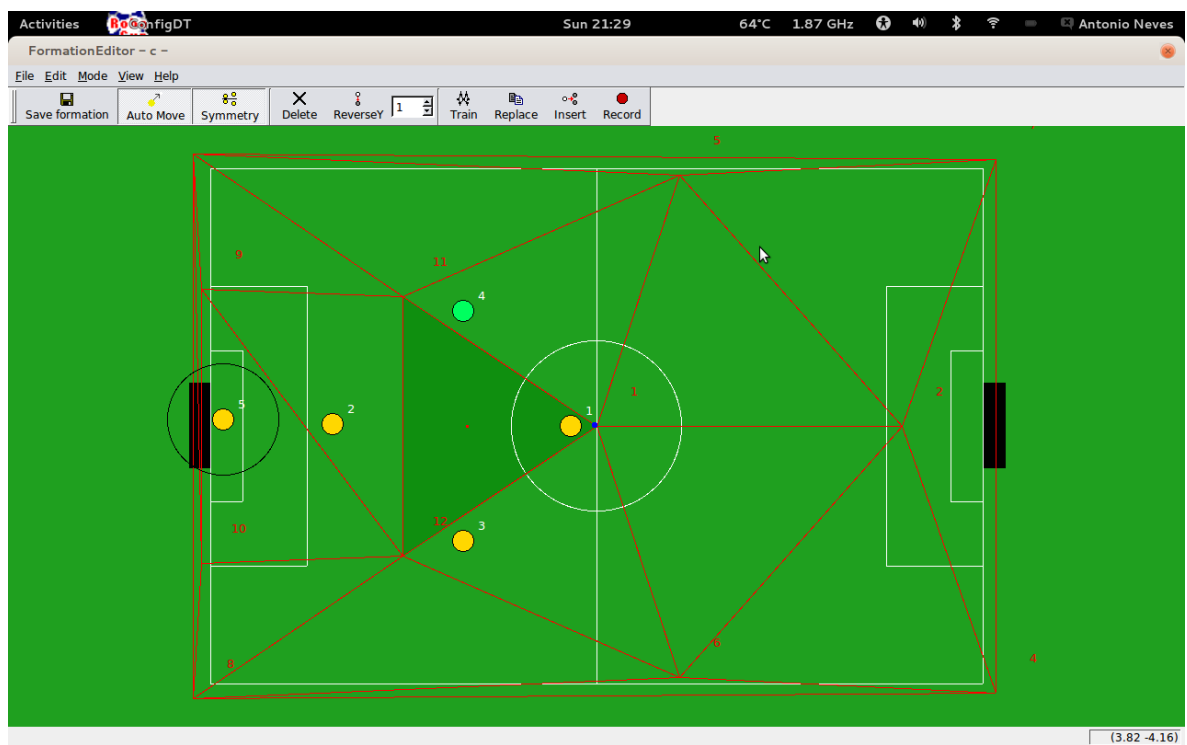


Figura 4.3: Desafio Científico - Formação Default

4.1.2 Classificação

A votação das equipas teve lugar após todas terem apresentado os seus trabalhos. O Apêndice C mostra os resultados obtidos pelas equipas, publicado pelo comité técnico do RoboCup, já com os resultados multiplicados pelo fator de cada equipa.

A classificação que foi obtida trouxe para a Universidade de Aveiro o 3º lugar neste desafio. Podemos na Tabela 4.1 ver os resultados finais ordenados da primeira até à última posição.

Equipa	Total	Posição
Teck United Eindhoven	427,46	1º
Carpe Noctem	425,82	2º
CAMBADA	414,93	3º
SocRob	375,21	4º
NuBot	373,34	5º
Hibikino-Musashi	336,07	6º
MRL	301,05	7º
BITAC	221,67	8º
Water	211,74	9º
IsePorto	0	10º

Tabela 4.1: Classificação final do Desafio Científico do RoboCup 2013

Capítulo 5

Conclusões

A criação do novo *Coach* revelou-se um trabalho criativo em toda a sua conceção. Não existindo restrições ao trabalho que podia ser desenvolvido apenas se ficou à mercê da criatividade, capacidade de análise das opções possíveis com a plataforma já existente do CAMBADA, facilidade de uso por parte da equipa e tempo de finalização a fim de poder ser apresentado no RoboCup 2013.

O trabalho desenvolvido consistiu na definição de um novo modelo de treinador, através da implementação de um novo agente designado *Coach*, que utiliza um conjunto de regras para a escolha da formação a usar pela equipa num determinado momento de jogo. A configuração destas regras fica a cargo de uma ferramenta gráfica, designada por *CoachConfig*, que gera um ficheiro XML que será depois interpretado pelo *Coach*. Houve, também, a necessidade de alterar algumas ferramentas já existentes no projeto para que este novo treinador fosse usado. Estas foram as principais contribuições deste trabalho.

O resultado final é um software fechado, completo e 100% funcional. Com provas dadas numa competição internacional, onde não só alcançou o 3^o lugar num desafio onde era focado isoladamente, como também ajudou a equipa a alcançar o 3^o lugar no torneio de equipas da MSL do RoboCup 2013. Ainda sobre os resultados obtidos do Desafio Científico do RoboCup 2013 teria sido muito mais proveitoso para análise deste trabalho os pontos que as equipas atribuíram ao projeto segundo os critérios. Caso as classificações dos critérios por equipa fossem revelados teríamos não só uma classificação quantitativa como também uma classificação qualitativa. Não obstante deste fator um 3^o lugar numa competição mundial é bastante gratificante.

O estado em que se encontra o novo *Coach* é já sem dúvida uma boa ferramenta para a equipa, porém deverá ser apenas um bom ponto de partida. Muito trabalho deverá ser feito no sentido de definir uma utilização proveitosa das novas funcionalidades que o novo *Coach* trás ao projeto CAMBADA, onde questões como quais as formações a usar e quando as usar ainda estão por analisar e concluir. Quando estas questões tiverem respostas mais concretas e fundamentadas o uso do novo *Coach* durante os jogos tornar-se-á inquestionável.

Também à medida que a capacidade técnica da equipa venha a aumentar no futuro o *Coach* deverá ser expandido, desta forma será possível criar mais tipos de regras e mais variáveis de decisão estatística. Uma das capacidades do novo *Coach* que mais desejo suscitou foi a capacidade de definir formações para quando a bola estiver na posse do adversário. Caso a equipa já conseguisse definir com alguma precisão a posição dos inimigos e se estes tinham

a bola em sua posse, teríamos assim a capacidade de definir formações de defesa com maior precisão [40].

Uma das características que foi ponderada foi a integração com as ferramentas de configuração já existentes no CMBADA. Optou-se por separar as ferramentas pois o tempo de desenvolvimento poderia ser comprometido com detalhes integração.

Sem dúvida que a maior dificuldade deste trabalho foi a realização de testes. Apesar de existir um simulador, este não simula com rigor a camada de comunicação, existiu sempre uma margem de desconfiança visto os testes serem rudimentares e praticamente todos no simulador. O trabalho foi realizado durante uma fase de mudança de plataforma no CMBADA o que impossibilitou testes reais nos robôs até ao início do RoboCup 2013, data em que a nova plataforma ficou funcional.

Caso o uso de regras se mantenha este poderá ser aumentado. Poderão vir a existir regras mais complexas, oriundas por exemplo duma abordagem de *Reinforcement Learning* ou de bases de dados processadas fora do tempo de jogo. A forma como as regras foram definidas teve em conta a facilidade de se expandirem as regras em trabalhos futuros.


```
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="Value" type="xs:double" use="required" />
  <xs:attribute name="StartTime" type="xs:duration" use="required" />
  <xs:attribute name="EndTime" type="xs:duration" use="required" />
  <xs:attribute name="DTFormation" type="xs:int" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="WindowDistMapSizeInSec" type="xs:integer" />
<xs:element name="DefaultFormation" type="xs:integer" />
<xs:element name="FightDistance" type="xs:double" />
<xs:element name="AwayFightTime" type="xs:double" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Apêndice B

Ficheiro de configuração usado no desafio científico do Robocup 2013

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ←
  xsi:noNamespaceSchemaLocation="CoachConfigFile.xsd">

  <Formations>
    <FormationRef ID="0" Name="default" Path="/home/an/old/research/_Robotics/cambada/←
      cambada2/config/DTs/c.conf" />
    <FormationRef ID="1" Name="A" Path="/home/an/old/research/_Robotics/cambada/←
      cambada2/config/DTs/a.conf" />
    <FormationRef ID="2" Name="B" Path="/home/an/old/research/_Robotics/cambada/←
      cambada2/config/DTs/b.conf" />
  </Formations>

  <Rules>
    <RuleConfig DTFormation="1" EndTime="PT5M" Follow="Both" ID="0" RuleType="←
      LargeLosing" StartTime="P0Y" Value="2" />
    <RuleConfig DTFormation="2" EndTime="PT5M" Follow="Both" ID="1" RuleType="←
      LargeWining" StartTime="P0Y" Value="2" />
    <RuleConfig DTFormation="2" EndTime="PT10M" Follow="Both" ID="2" RuleType="←
      LessInOurFieldTime" StartTime="PT5M" Value="50" />
    <RuleConfig DTFormation="1" EndTime="PT10M" Follow="Both" ID="3" RuleType="←
      LessInTheirFieldTime" StartTime="PT5M" Value="50" />
    <RuleConfig DTFormation="2" EndTime="PT15M" Follow="Defence" ID="4" RuleType="←
      DontCare" StartTime="PT10M" Value="1" />
    <RuleConfig DTFormation="1" EndTime="PT15M" Follow="Attack" ID="5" RuleType="←
      DontCare" StartTime="PT10M" Value="1" />
  </Rules>

  <WindowDistMapSizeInSec>60</WindowDistMapSizeInSec>
  <DefaultFormation>0</DefaultFormation>
  <FightDistance>0.1</FightDistance>
  <AwayFightTime>5</AwayFightTime>
</Config>
```


Apêndice C

Tabela de Classificações do Desafio Científico do RoboCup 2013

As informações que se seguem foram recriadas a partir de uma fotografia. As classificações detalhadas do Desafio Científico do RoboCup 2013, foram publicadas durante o evento do RoboCup. A fotografia em questão foi adicionada no Apêndice D.

Equipa	Pontos atribuídos pela equipa										Pré-total
	MRL	Water	IsePorto	BITAC	CAMBADA	NuBot	kibikino-musashi	SocRob	Teck United Eindhoven	Carpe Noctem	
MRL		35	42	35	20	33	45	33	30	36	309
Water	5		29	30	14	29	33	30	30	18	218
IsePorto	0	0		0	0	0	0	0	0	0	0
BITAC	11	29	13		17	24	32	37	36	24	223
CAMBADA	44	45	44	44		44	50	45	48,5	53	417,5
NuBot	46	38	33	38	47		40	44	38	44	368
kibikino-musashi	19	34	47	40	38	39		40	38,5	38	333,5
SocRob	39	32	39	42	38	45	42		46,5	48	371,5
Teck United Eindhoven	46	44	41	47	52	52	49	42		49	422
Carpe Noctem	45	39	49	44	50	41	58	50	48		424

Pontuação Média 25,50 29,60 33,70 32,00 27,60 30,70 34,90 32,10 31,55 31,00

Média das P. Médias 30,87

Fator de Correção 1,21 1,04 0,92 0,96 1,12 1,01 0,88 0,96 0,98 1,00

Resultados Finais

Equipa	MRL	Water	IsePorto	BITAC	CAMBADA	NuBot	kibikino-musashi	SocRob	Teck United Eindhoven	Carpe Noctem	Total
MRL		36,40	38,64	33,60	22,40	33,33	39,60	31,68	29,40	36,00	301,05
Water	6,05		26,68	28,80	15,68	29,29	29,04	28,80	29,40	18,00	211,74
IsePorto	0,00	0,00		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0
BITAC	13,31	30,16	11,96		19,04	24,24	28,16	35,52	35,28	24,00	221,67
CAMBADA	53,24	46,80	40,48	42,24		44,44	44,00	43,20	47,53	53,00	414,93
NuBot	55,66	39,52	30,36	36,48	52,64		35,20	42,24	37,24	44,00	373,34
kibikino-musashi	22,99	35,36	43,24	38,40	42,56	39,39		38,40	37,73	38,00	336,07
SocRob	47,19	33,28	35,88	40,32	42,56	45,45	36,96		45,57	48,00	375,21
Teck United Eindhoven	55,66	45,76	37,72	45,12	58,24	52,52	43,12	40,32		49,00	427,46
Carpe Noctem	54,45	40,56	45,08	42,24	56,00	41,41	51,04	48,00	47,04		425,82

Apêndice D

Fotografia das Classificações do Desafio Científico do RoboCup 2013

RoboCup 2013, Eindhoven
Middle-Size League
Scientific Challenge - Results

RAW RESULTS

Team	Points given by each team to the others										Total
	MRL	Water	IsePorto	BITAC	CAMBADA	NuBot	Hibikino-Musashi	SocRob	Tech United Eindhoven	Carpe Noctem	
MRL	0	35	42	35	20	33	45	33	30	36	309
Water	5	0	29	30	14	29	33	30	30	18	218
IsePorto	0	0	0	0	0	0	0	0	0	0	0
BITAC	11	29	13	0	17	24	32	37	36	24	223
CAMBADA	44	45	44	44	0	44	50	45	48,5	53	417,5
NuBot	46	38	33	38	47	0	40	44	38,5	44	368
Hibikino-Musashi	19	34	47	40	38	39	0	40	38,5	38	333,5
SocRob	39	32	39	42	38	45	42	0	46,5	48	371,5
Tech United Eindhoven	46	44	41	47	52	52	49	42	0	49	422
Carpe Noctem	45	39	49	44	50	41	58	50	48	0	424
	255	296	337	320	276	307	349	321	315,5	310	

Team Leader's Average (TLav)	25,50	29,60	33,70	32,00	27,60	30,70	34,90	32,10	31,55	31,00
Standard Average (Clav)	30,87									
Correction Factor (Cr)	1,21	1,04	0,92	0,96	1,12	1,01	0,88	0,96	0,98	1,00

FINAL RESULTS (corrected)

Team	MRL	Water	IsePorto	BITAC	CAMBADA	NuBot	Hibikino-Musashi	SocRob	Tech United Eindhoven	Carpe Noctem	Total
MRL	0,00	36,40	38,64	33,60	22,40	33,33	39,60	31,68	29,40	36,00	301,05
Water	6,05	0,00	26,68	28,80	15,68	29,29	29,04	28,80	29,40	18,00	211,74
IsePorto	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BITAC	13,31	30,16	11,96	0,00	19,04	24,24	28,16	35,52	35,28	24,00	221,67
CAMBADA	53,24	46,80	40,48	42,24	0,00	44,44	44,00	43,20	47,53	53,00	414,93
NuBot	55,66	39,52	30,36	36,48	52,64	0,00	35,20	42,24	37,24	44,00	373,34
Hibikino-Musashi	22,99	35,36	43,24	38,40	42,56	39,39	0,00	36,96	45,57	48,00	375,21
SocRob	47,19	33,28	35,88	40,32	42,56	45,45	36,96	0,00	45,57	48,00	375,21
Tech United Eindhoven	55,66	45,76	37,72	45,12	58,24	52,52	43,12	40,32	0,00	49,00	427,46
Carpe Noctem	54,45	40,56	45,08	42,24	56,00	41,41	51,04	48,00	47,04	0,00	425,82

Apêndice E

Slides da apresentação do novo *Coach* no Desafio Científico do RoboCup 2013






Dynamic Strategy Positioning Using an Autonomous Coach



IEETA - DETI
 University of Aveiro
 Portugal

CAMBADA Free Challenge, RoboCup 2013

1






Outline

- Motivation
- Proposed autonomous coach for dynamic formations in CAMBADA
- Demo

CAMBADA Free Challenge, RoboCup 2013

2






Motivation

- Coordination challenges: Role assignment, Formation control, Plan execution, Communication, ...
- Formations are an essential concept in a robotic soccer strategy
 - Provide a coordination framework
 - Have a real impact on the team performance
 - Can/should be adapted to the team and opponent capabilities
 - **DT - Delaunay Triangulation formations**
 - (Akiyama et al., 2007)
 - Added flexibility in the definition of positionings
 - **Presented in the Free Challenge 2011 by CAMBADA**

CAMBADA Free Challenge, RoboCup 2013

3






Proposed autonomous coach

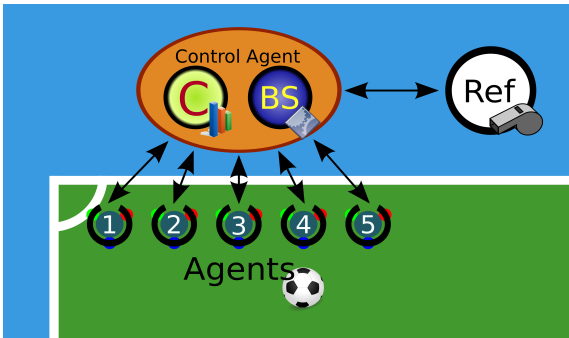
- **GOAL:**
Depending on the conditions of the game (goals, ball possession, ball position, time, ...) decide what is the strategy to be used by the team.
- **HOW:**
Using an autonomous coach that is able to process rules based on information shared by the field agents and current state of the game.

CAMBADA Free Challenge, RoboCup 2013

4






Proposed autonomous coach

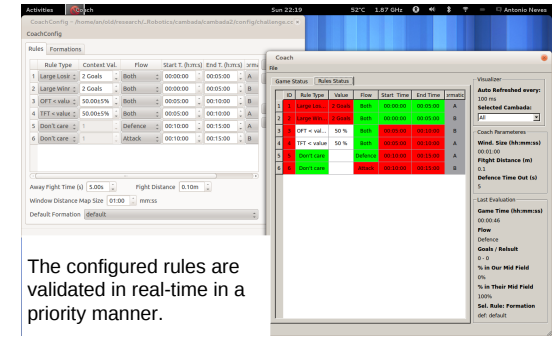


CAMBADA Free Challenge, RoboCup 2013

5


Proposed autonomous coach



The configured rules are validated in real-time in a priority manner.

CAMBADA Free Challenge, RoboCup 2013

6



Proposed autonomous coach



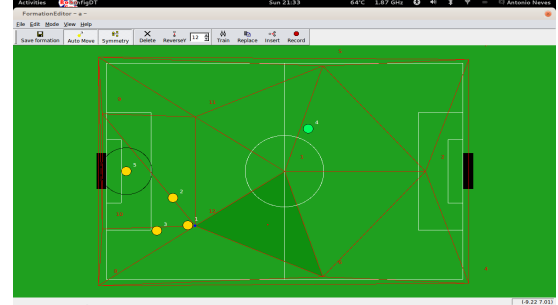
Example of the gathered statistics about the ball position.

CAMBADA Free Challenge, RoboCup 2013

7


Demo: used formations (A)

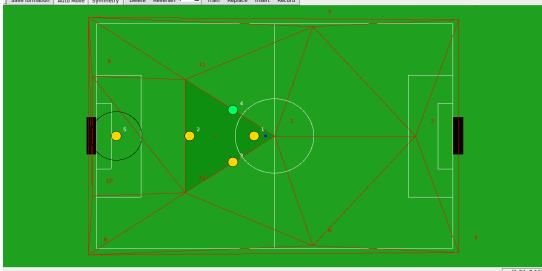


Loosing situation, ball more time in our field and our possession

CAMBADA Free Challenge, RoboCup 2013


8

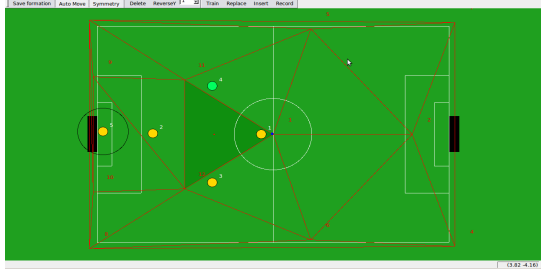
Demo: used formations (B) 




Winning situation, ball more time in their field and their possession

CAMBADA Free Challenge, RoboCup 2013 9

Demo: used formations (default) 



CAMBADA Free Challenge, RoboCup 2013 10

Demo: used formations (default) 

DEMO

CAMBADA Free Challenge, RoboCup 2013 11

Bibliografia

- [1] CAMBADA homepage. <http://robotica.ua.pt/CAMBADA/index.htm>. Accessed Jul 15, 2013.
- [2] Automated guided vehicles - agv. <http://www.terex.com>. Accessed Nov 23, 2013.
- [3] CAMBADA Robotic Soccer - University of Aveiro. <http://robotica.ua.pt/CAMBADA/pictures-2011.htm>. Accessed: 2013-11-23.
- [4] Codesynthesis xsd: Xml data binding for c++. <http://www.codesynthesis.com/products/xsd/>. Accessed Aug 11, 2013.
- [5] Dance - RoboCupJunior. <http://rcj.robocup.org/dance.html>. Accessed: 2013-11-23.
- [6] Darmstadt Dribblers dominam a Humanoid League. http://www.tu-darmstadt.de/vorbeischaue/aktuell/archiv_2/neuesausdertudeinzelansicht_47552.de.jsp. Accessed: 2013-11-24.
- [7] Institut für Automatik, IfA robocup. <http://control.ee.ethz.ch/index.cgi?page=sada;action=details;id=363>. Accessed: 2013-11-24.
- [8] NimbRo @Home - Autonomous Intelligent Systems. <http://www.nimbro.net/@Home/index.html>. Accessed: 2013-11-23.
- [9] Qt framework oficial site. <http://qt.digia.com>. Accessed Jun 8, 2013.
- [10] Refbox oficial site. <http://msl-refbox.sourceforge.net>. Accessed Jul 19, 2013.
- [11] Rescue - RoboCupJunior. <http://rcj.robocup.org/rescue.html>. Accessed: 2013-11-23.
- [12] RoboCup @ Work - RoboCup German Open 2014. <http://www.robocupgermanopen.de/en/major/atwork>. Accessed: 2013-11-23.
- [13] RoboCup 2013 Rescue Simulation League Results - RoboCup Rescue Simulation League. <http://roborescue.sourceforge.net/2013/results/Istanbul3/index.html>. Accessed: 2013-10-13.
- [14] RoboCup Eindhoven 2013. <http://www.robocup2013.org/>. Accessed: 2013-8-2.
- [15] Robocup events page. <http://www.robocup.org/events/events-2000/>. Accessed Oct 12, 2013.

- [16] RoboCup Junior Dance - RoboCup German Open 2014. <http://www.robocupgermanopen.de/en/junior/dance>. Accessed: 2013-11-23.
- [17] RoboCup Junior Rescue - RoboCup German Open 2014. <http://www.robocupgermanopen.de/en/junior/rescue>. Accessed: 2013-11-23.
- [18] RoboCup Junior Soccer - RoboCup German Open 2014. <http://www.robocupgermanopen.de/en/junior/soccer>. Accessed: 2013-11-23.
- [19] RoboCup Logistics League. <http://www.robocup-logistics.org/>. Accessed: 2013-11-23.
- [20] Robocup technical committee: Robocup standard platform league (nao) rule. <http://www.tzi.de/spl/pub/Website/Downloads/Rules2013.pdf>. Accessed: 2013-11-2.
- [21] Soccer - RoboCupJunior. <http://rcj.robocup.org/soccer.html>. Accessed: 2013-11-23.
- [22] Volvo autonomous road train hits public roads for first time. <http://wot.motortrend.com/volvo-autonomous-road-train-hits-public-roads-for-first-time-211941.html/volvo-autonomous-road-train-sartre-project-info-graphic-7/>. Accessed: 2013-11-24.
- [23] Will Robots Replace Humans? audio/video index. <http://www.innovations.gatech.edu/robotics/avindex.php>. Accessed: 2013-11-24.
- [24] ¡Hola! Mexico: RoboCup 2012 calls for full commitment - Initiative Mittelstand. <http://www.robocupgermanopen.de/en/major/atwork>. Accessed: 2013-11-23.
- [25] Hidehisa Akiyama and Itsuki Noda. Multi-agent positioning mechanism in the dynamic environment. In *RoboCup 2007: Robot Soccer World Cup XI*. Springer, 2008.
- [26] Luis Almeida, Frederico Santos, Tullio Facchinetti, Paulo Pedreiras, Valter Silva, and L Seabra Lopes. Coordinating distributed autonomous agents with a real-time database: The cambada project. In *Computer and Information Sciences-ISCIS 2004*, pages 876–886. Springer, 2004.
- [27] José Luís Azevedo, Bernardo Cunha, and Luís Almeida. Hierarchical distributed architectures for autonomous mobile robots: a case study. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 973–980. IEEE, 2007.
- [28] Andy Baker and Ian Mackenzie. Omnidirectional drive systems kinematics and control. In *1st Robotics Conference*, 2008.
- [29] Guilherme Bittencourt. Inteligência artificial distribuída. *UFSC-http://www.lcmi.ufsc.br/gia/ic-artigos/iad. ps. gz*, 1998.
- [30] Eric Chan, Peter Gilhead, Pavel Jelínek, Petr Krejčí, and Tom Robinson. Cooperative control of sartre automated platoon vehicles.
- [31] George Coulouris, Jean Dollimore, and Tim Kindberg. Distributed systems: Concepts and design edition 3. *paragraph*, 15:04–04, 2001.

- [32] Augusto Pereira da Rosa. Pré-história: Educação para sobrevivência. *Maiêutica-Artes Visuais*, 1(1), 2013.
- [33] Nuno M Figueiredo, António JR Neves, Nuno Lau, Artur Pereira, and Gustavo Corrente. Control and monitoring of a robotic soccer team: The base station application. In *Progress in Artificial Intelligence*, pages 299–309. Springer, 2009.
- [34] Lisa Jean Moya and Andreas Tolk. Towards a taxonomy of agents and multi-agent systems. In *Proceedings of the 2007 spring simulation multiconference- Volume 2*. Society for Computer Simulation International, 2007.
- [35] AJR Neves, JL Azevedo, B Cunha, J Cunha, R Dias, P Fonseca, N Lau, J Silva, T Ângelo, C Cruz, G Corrente, LS Lopes, P Pedereiras, A Pereira, A Pinho, J Rodrigues, F Santos, and J Vieira. Cambada’2011: Team description paper. *Proceedings Robocup 2011*, 2011.
- [36] AJR Neves, JL Azevedo, MB Cunha, N Lau, A Pereira, G Corrente, F Santos, D Martins, N Figueiredo, J Silva, et al. Cambada’2010: Team description paper. *Proceedings Robocup 2010*, 2010.
- [37] Luís Paulo Gonçalves dos Reis et al. Coordenação em sistemas multi-agente: Aplicações na gestão universitária e futebol robótico. 2012.
- [38] Daniela Fouchard Severo, Lori Viali, and João Bernardes da Rocha Filho. O estudo da geometria plana e espacial a partir da construção de um caleidoscópio. *Contribuições de um Museu Interativo à Educação em Ciências e Matemática*, 2009.
- [39] Maarten Sierhuis, William J Clancey, Ron JJ van Hoof, Chin H Seah, Michael S Scott, Robert A Nado, Susan F Blumenberg, Michael G Shafto, Brian L Anderson, Anthony C Bruins, et al. Nasa’s oca mirroring system an application of multiagent systems in mission control. In *AAMAS*, volume 9, 2009.
- [40] Timo Steffens. Feature-based declarative opponent-modelling in multi-agent systems. 2002.
- [41] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2), 1999.
- [42] RoboCup Federation Wiki. @Home League, 2013.
- [43] RoboCup Federation Wiki. Humanoid league, 2013.
- [44] RoboCup Federation Wiki. Middle size league, 2013.
- [45] RoboCup Federation Wiki. Rescue simulation league, 2013.
- [46] RoboCup Federation Wiki. Robot league, 2013.
- [47] RoboCup Federation Wiki. Small size league, 2013.
- [48] RoboCup Federation Wiki. Soccer simulation league, 2013.

- [49] Wikipedia. Delaunay triangulation — wikipedia, the free encyclopedia, 2013. [Online; accessed 23-November-2013].
- [50] Wikipedia. Raúl rojas — wikipedia, the free encyclopedia, 2013. [Online; accessed 24-November-2013].
- [51] Wikipedia. Rescue robot league — wikipedia, the free encyclopedia, 2013. [Online; accessed 24-November-2013].
- [52] Wikipedia. Robocup 2d soccer simulation league — wikipedia, the free encyclopedia, 2013. [Online; accessed 24-November-2013].
- [53] Wikipedia. Swarm robotics — wikipedia, the free encyclopedia, 2013. [Online; accessed 24-November-2013].