Capítulo 8

8. Análise do Domínio de Aplicação: Futebol Robótico

Dos domínios de aplicação seleccionados para testar as estratégias de coordenação de agentes desenvolvidas, o Futebol Robótico com ênfase na liga de simulação, é aquele a que vamos dar maior relevância. A iniciativa RoboCup [Kitano et al., 1995] [Kitano, 1997] é um projecto de investigação e educação internacional com o objectivo de promover a investigação em Inteligência Artificial (Distribuída) e Robótica Inteligente [RoboCup, 2001]. O projecto baseia-se na utilização de um problema standard – o futebol robótico – onde um elevado conjunto de tecnologias é necessário para possibilitar a construção de uma equipa de Robôs reais ou virtuais capaz de participar num desafio de futebol, jogado de acordo com um determinado conjunto de regras pré-especificado.

O futebol robótico inclui diversas ligas que se dividem em dois tipos: ligas robóticas (utilizando robôs pequenos, médios, cães e humanóides) e a liga de simulação. Cada liga coloca um conjunto de desafios de investigação próprio e tem ênfase em determinados tópicos necessários para efectivamente colocar equipas de robôs a disputar uma partida de futebol. Por exemplo, na liga de simulação, a ênfase é colocada na coordenação em SMA, enquanto na liga de robôs pequenos, a ênfase é colocada no controlo rápido e preciso dos robôs e na liga de robôs médios os tópicos mais importantes incluem a visão computacional, projecto electromecânico e auto-localização dos robôs.

Existem também outros desafios associados ao futebol robótico que a iniciativa RoboCup promove. De entre estes destacam-se o *RoboCup Rescue* [Kitano et al., 1999] e o *RoboCup Júnior* [Sklar et al., 2002]. O *Rescue* tem como objectivo estimular a aplicação da investigação realizada no futebol robótico, a domínios socialmente mais úteis, no caso, missões de salvamento e resgate em grandes catástrofes [Kitano et al., 1999] [Rescue, 2001]. Divide-se em duas ligas: robótica e simulação. O *RoboCup Júnior* surgiu como uma forma de estimular os mais jovens a participar no *RoboCup* [Sklar et al., 2002]. Providencia as condições que permitem a jovens em idade escolar, construir e colocar em funcionamento os seus robôs para realizar diversas tarefas.

Neste trabalho um destaque especial é dado à liga de simulação do *RoboCup*. Esta liga é baseada no sistema de simulação *soccerserver* [Noda, 1995] [Noda et a., 1998] que permite a duas equipas de agentes autónomos disputar um jogo de futebol simulado. O *soccerserver* providencia um domínio muito realístico, no sentido em que inclui muitas complexidades do futebol real e dos sistemas robóticos tais como erros nos sensores e actuadores e energia limitada [Noda e Stone, 2002].

Este capítulo é importante para perceber o trabalho apresentado no capítulo nove relativo à proposta de metodologias de coordenação para equipas de agentes. A complexidade geral do RoboCup e em particular do *soccerserver* implica a necessidade de uma descrição prévia do concurso e da liga de simulação, introduzindo os desafios de investigação e terminologia própria do sistema *soccerserver* para que possa ser compreendido o trabalho realizado utilizando-o como plataforma de teste.

Este capítulo está organizado da seguinte forma. A secção 8.1 apresenta uma descrição da iniciativa *RoboCup*, dos seus objectivos e história. Na secção 8.2 é apresentada uma panorâmica sobre as diversas ligas do *RoboCup* e os desafios de investigação que colocam. Na secção 8.3 é apresentado o futebol robótico simulado e é detalhado o funcionamento do simulador (*soccerserver*) que permite realizar jogos de futebol robótico simulado. A secção seguinte descreve resumidamente a investigação realizada por diferentes grupos de investigação utilizando os problemas colocados pelo RoboCup. Em seguida é apresentado o trabalho relacionado e são descritas outras aplicações da investigação realizada no âmbito da liga de simulação do *RoboCup*. O capítulo conclui-se com a discussão das vantagens e perigos da realização de competições científicas como o RoboCup e com algumas conclusões sobre este domínio.

8.1 Introdução

8.1.1 Objectivos do RoboCup

O objectivo da iniciativa *RoboCup* é promover a investigação em Robótica Inteligente e Inteligência Artificial através de um desafio que seja simultaneamente estimulante do ponto de vista científico, colocando um vasto conjunto de problemas científicos aos investigadores da área mas, ao mesmo tempo, extremamente atraente para o público em geral e para os meios de comunicação social. Com esta vertente, pretende-se que o *RoboCup* seja capaz de chamar a atenção do público e comunicação social, não só para o *RoboCup* como competição por si só, mas também para a investigação realizada em Inteligência Artificial e Robótica Inteligente pelos diversos laboratórios participantes.

Como forma de promover a investigação na área, foi lançado um objectivo de longo prazo:

"No ano de 2050, uma equipa de robôs autónomos humanóides, ser capaz de vencer a equipa campeã do mundo de futebol, num encontro disputado de acordo com as regras da FIFA." [Kitano, 1997]

Este objectivo é actualmente partilhado como um dos grandes desafios da comunidade, pelos investigadores da área da Inteligência Artificial e Robótica a atingir durante os próximos 50 anos. Embora este desafio, à luz da ciência e tecnologia actuais, pareça altamente ambicioso, a colocação de objectivos científicos bem definidos de longo prazo, tem sido, ao longo dos anos, uma forma de estimular o desenvolvimento científico [RoboCup, 2001]. Um exemplo marcante disto surgiu com o projecto *Apollo* e com o seu objectivo de longo prazo:

"*Aterrar um homem na Lua e fazê-lo regressar, são e salvo à Terra"* [Kennedy, 1961].

Embora o alcance deste objectivo de longo prazo tenha constituído um marco na história da humanidade, por si só não ofereceu um grande impacto económico ou social. As tecnologias desenvolvidas para conseguir alcançar este objectivo formaram uma base sólida para enormes progressos económicos e sociais. No caso do projecto *Apollo*, a motivação dada pela ambição e abrangência do objectivo final, deu origem a avanços tecnológicos e científicos que foram a base das fundações humanas e tecnológicas da poderosa indústria Americana dos dias de hoje [RoboCup, 2001].

No projecto *Apollo*, embora o objectivo fosse aterrar um homem na Lua, a sua abrangência era muito maior. Neil Armstrong após aterrar na Lua, afirmou: *"É um pequeno passo para o homem, mas um salto gigante para a humanidade!"*. Esta frase mostrou o cumprimento do objectivo inicial, mas os objectivos do projecto *Apollo* iam muito para além de aterrar um humano na Lua: Desenvolver tecnologia que permitisse estabelecer interesses Americanos no espaço; dar a proeminência no Espaço aos EUA; desenvolver um programa alargado de pesquisa na Lua; desenvolver a capacidade humana para trabalhar na Lua e em outros planetas, etc.

Embora o objectivo de longo prazo do *RoboCup* seja o desenvolvimento de uma equipa capaz de vencer o campeão do mundo humano de futebol, outros objectivos mais modestos podem ser estabelecidos com base neste, como por exemplo, conseguir criar uma equipa de robôs que jogue tal como uma equipa de humanos ou, criar robôs capazes de jogar um jogo de futebol contra um equipa de humanos. Para além disto, o objectivo final da iniciativa *RoboCup*, embora impossível no curto-prazo em face da ciência e tecnologia actuais, irá criar um conjunto de sub-objectivos que levarão até ao objectivo final. Entre estes sub-objectivos, destaca-se a criação de equipas de robôs reais e virtuais, de diversos tipos e com características distintas, que sejam capazes de jogar razoavelmente com regras modificadas, no sentido de as simplificar, relativamente às da FIFA [FIFA, 2001].

Um outro desafio semelhante colocado aos investigadores em Inteligência Artificial no decurso das últimas 4 décadas, consistiu em construir um agente (programa) que fosse

capaz de vencer o campeão mundial de Xadrez utilizando as regras oficiais da Federação Internacional de Xadrez. Este desafio para além de promover a investigação em Inteligência Artificial, mostrou a importância da existência de problemas *standard* em que diferentes metodologias e avanços científicos podem ser comparados. Esta faceta de problema *standard* foi extremamente importante no Xadrez. Diversos algoritmos de pesquisa, arquitecturas de computadores e metodologias científicas foram desenvolvidos para este domínio. Em Maio de 1997, o computador Deep Blue da IBM [Deep Blue, 1997] derrotou Gary Kasparov (o campeão humano de Xadrez), utilizando as regras oficiais do Xadrez [Deep Blue, 1997]. Com esta vitória, o desafio de 40 anos da Inteligência Artificial utilizando o Xadrez como domínio de aplicação ficou muito próximo de um final com sucesso.

Uma das características que tornou o Xadrez computorizado um problema *standard* foi a facilidade que este domínio ofereceu para a comparação de abordagens distintas, e para a avaliação do progresso científico global realizado no domínio. No entanto, com a chegada ao final do desafio associado ao Xadrez, no mundo da IA, novos domínios e problemas *standard* mais complexos e estimulantes tornaram-se necessários. Foi neste contexto que se desenvolveu o desafio do futebol robótico como um problema *standard* para a IAD e robótica inteligente.

Uma característica essencial do RoboCup consiste em visualizá-lo também como um problema *standard* em que várias teorias, arquitecturas, metodologias e algoritmos podem ser avaliadas e facilmente comparadas entre si. O domínio preenche inteiramente as características necessárias para ser o sucessor do Xadrez como problema *standard* da IA e domínio para a comparação de diversas metodologias da IA e Robótica [Reis, 2001a]. As diferenças principais entre o domínio do Xadrez e o *RoboCup* podem ser visualizadas na tabela 9.

Domínio / *Característica*	***Xadrez***	***RoboCup***
Ambiente	Estático	Dinâmico
Mudança de Estado	Por turnos	Tempo-Real
Acessibilidade ao Estado do Mundo	Completa	Incompleta
Resultado das Acções	Determinístico	Não Determinístico
Leitura dos Sensores	Discreta (Simbólica)	Contínua (não simbólica)
Utilização dos Actuadores	Discreta (Simbólica)	Contínua (não simbólica)
Controlo	Centralizado	Distribuído

Tabela 9: Diferenças entre as características dos domínios do RoboCup e Xadrez.

Analisando a tabela 9 é visível que o *RoboCup* constitui um domínio que inclui características bastante mais complexas que o Xadrez. Uma das diferenças principais reside no facto do seu ambiente ser dinâmico. Enquanto os agentes (jogadores virtuais ou robóticos) decidem a próxima acção a executar, a bola encontra-se em movimento, assim como os agentes colegas de equipa e os agentes adversários. No Xadrez, todos os objectos estão estacionários enquanto os agentes decidem a próxima acção a executar.

No Xadrez a mudança de estado é realizada por turnos (jogadas alternadas de ambos os jogadores). No *RoboCup* todos os jogadores (de ambas as equipas) podem agir ao mesmo tempo no mundo.

A acessibilidade ao estado do mundo é completa no Xadrez. No *RoboCup* a acessibilidade ao estado do mundo é dada unicamente através dos sensores de cada robô real ou virtual. A capacidade destes sensores é limitada, pelo que só uma parcela muito reduzida do estado do mundo se encontra acessível a cada momento aos robôs. Por exemplo, no caso da visão, quer o seu alcance quer o seu ângulo de abertura são limitados, conduzindo a que unicamente uma parcela do campo, usualmente muito reduzida, seja visível em cada instante. Para obter uma perspectiva global do campo e do posicionamento dos diversos objectos neste, é necessário procurar construir um estado do mundo global (representação interna do conhecimento sobre o estado do mundo, que integre os valores recebidos dos sensores no passado com os valores actuais) partindo das visões parciais obtidas em cada instante.

Outra complexidade adicional do ambiente do *RoboCup* consiste no facto de as acções efectuadas pelos agentes não terem um efeito único e garantido, ou seja, o ambiente não ser determinístico. No xadrez a realização de uma dada jogada tem um efeito único e garantido e não existe qualquer possibilidade de falha. Ao contrário, no futebol robótico, a realização de uma dada acção, por exemplo chutar a bola com uma dada potência numa dada direcção, não tem um efeito totalmente previsível. Pode acontecer que o agente nem sequer consiga chutar a bola. Mesmo no caso de o chuto ser bem sucedido, a potência e direcção do chuto efectivamente aplicadas à bola não são totalmente previsíveis.

O estado do mundo, a leitura dos sensores e a utilização dos actuadores são compostos por variáveis discretas no Xadrez. No entanto, no *RoboCup*, todas estas variáveis são contínuas, tornando assim este domínio bastante mais complexo. O número de percepções possível, estados admissíveis do mundo e acções possíveis de executar são, em cada instante, virtualmente infinitas no RoboCup. Contribuindo também para este aumento de complexidade, encontra-se o facto de os valores serem lidos pelos sensores, incluindo erros consideráveis e as ordens enviadas aos actuadores serem também executadas com erros significativos.

A diferença final entre os domínios consiste no facto de o controlo no *RoboCup* ser em geral distribuído. Dependendo da modalidade, um número elevado de agentes autónomos

(que pode variar entre 4 e 11 nas ligas de futebol robótico), tem de agir autonomamente e coordenarem-se de forma a atingir um objectivo comum.

O *RoboCup* foi desta forma projectado de forma a colocar num mundo limitado, um conjunto elevado de complexidades do mundo real, mantendo no entanto o custo, complexidade global e dimensão do problema, acessível aos grupos de investigação em Robótica e Inteligência Artificial. Os problemas de investigação colocados pelo *RoboCup* de uma forma integrada, cobrem uma vasta área dos domínios da IA e Robótica, incluindo: coordenação, cooperação e comunicação multi-agente, arquitecturas de agentes inteligentes, aprendizagem, planeamento em tempo-real, decisão estratégica e táctica, comportamento reactivo, visão por computador, processamento e análise de imagem, sistemas de locomoção e actuação, sistemas sensoriais, fusão sensorial em tempo-real, navegação, controlo inteligente robótico, etc.

8.1.2 História do RoboCup

A ideia do futebol robótico foi introduzida por Alan Mackworth da Universidade de British Columbia, Canadá em 1992[41] [Mackworth, 1993]. De forma independente, um grupo de investigadores Japoneses de IA, organizaram um *Workshop* intitulado "Os Grandes Desafios da IA" em Outubro de 1992 em Tóquio. Neste *Workshop* foram discutidas diversas questões sobre a utilização do futebol na promoção da investigação realizada em Robótica e IA, realizado um estudo financeiro, tecnológico e do impacto social desta utilização. Foram também definidos protótipos de robôs futebolistas e um primeiro projecto de um simulador de futebol. O resultado do estudo foi o lançamento em Junho de 1993 de uma competição robótica intitulada *Robot J-League*[42] que após as excelentes reacções de grupos de investigação fora do Japão, nomeadamente grupos de investigação Americanos, foi estendida para um projecto internacional conjunto com o nome *Robot World Cup Initiative – RoboCup*.

Paralelamente a esta discussão, diversos investigadores estavam já a utilizar o futebol robótico como domínio da sua investigação. Itsuki Noda no ElectroTechnical Laboratory (ETL) do Japão estava a desenvolver um simulador de jogos virtuais de futebol que, posteriormente daria origem ao *soccerserver* [Soccerserver, 2001], o simulador oficial da liga de simulação do RoboCup. Minoru Asada e seus colaboradores na Universidade de Osaka e Manuela Veloso e Peter Stone na Carnegie Mellon University estavam também envolvidos na criação de robôs futebolistas.

---

[41] O artigo original "On Seeing Robots", embora datado de 1992, foi publicado posteriormente em 1993 no livro Computer Vision: Systems, Theory and Applications, pp. 1-13.

[42] A J-League é a liga profissional de futebol no Japão

Em Setembro de 1993, o primeiro anúncio oficial desta iniciativa foi realizado e regras específicas foram criadas para diversas ligas de futebol robótico. Itsuki Noda no ETL lançou a versão 0 do *SoccerServer* (desenvolvida em LISP), e depois a versão 1.0, desenvolvida em C++. A primeira demonstração pública deste sistema foi realizada na conferência IJCAI'95[43], que teve lugar em Agosto de 1995 em Montreal, Canadá. No decurso desta conferência foi também realizado o anúncio oficial da organização do primeiro *RoboCup – Robot World Cup Soccer Games and Conferences*, em conjunção com a IJCAI'97 em Nagoya no Japão. Ao mesmo tempo foi decidido realizar um Pre-RoboCup em 1996 para identificar e avaliar os possíveis problemas da organização do RoboCup97.

O Pre-RoboCup96 realizou-se durante a IROS'96[44] em Osaka em Novembro de 1996 com oito equipas na modalidade de simulação e uma demonstração de robôs médios. A primeira competição RoboCup realizou-se em Nagoya em 1997 com mais de 40 equipas participantes. Mais de 5 mil espectadores assistiram aos jogos [Robocup, 2001] num dos maiores eventos científicos de sempre.

Figura 45: Aspecto Geral do Fukuoka Dome – RoboCup2002

As cinco edições seguintes do *RoboCup* disputaram-se em Paris 1998, Estocolmo 1999, Melbourne 2000, Seattle 2001 e Fukuoka 2002. Para além destas competições, diversas outras competições regionais ou nacionais tiveram lugar. De entre estas destacam-se o *Japan Open* (disputado em 2000, 2001 e 2002), o campeonato Europeu (disputado unicamente em 2000 em Amesterdão), o *German Open* (2001 e 2002) e o *China Open* (2000, 2001 e 2002). As últimas edições do *RoboCup* transformaram-se em grandes

[43] IJCAI – International Joint Conference on Artificial Intelligence

[44] IROS – International Conference on Intelligence Robotics and Systems

eventos científicos com centenas de participantes e milhares de espectadores. No RoboCup 2002 em Fukuoka participaram 288 equipas, com mais de 1000 investigadores envolvidos e assistiram ao certame um total de mais de 110 mil espectadores (figura 45).

8.1.3 A Federação do RoboCup

A federação do *RoboCup* (*RoboCup Federation*) é uma associação internacional, registada na Suíça, que tem como principiais objectivos organizar o esforço internacional para a promoção da ciência e Tecnologia, utilizando jogos de futebol com robôs e agentes de *software* [RoboCup, 2001].

As funções principais da Federação *RoboCup* estão ligadas à organização anual de um campeonato do mundo e à coordenação dos esforços globais de todos os investigadores da área. Os membros da Federação são todos investigadores activos da área e representam um alargado número de Universidades e grandes empresas. Como o número de investigadores envolvido no projecto é muito alargado e encontra-se espalhado por todo o globo, diversos comités regionais estão formados para promover a investigação do *RoboCup* em áreas geográficas localizadas.

A federação é composta por um Presidente, um conjunto de *Trustees*, um comité executivo, um comité de aconselhamento e comités técnicos (compostos por sete elementos cada) para cada uma das ligas. Comités específicos são designados pela Federação para os eventos organizados tais como o campeonato mundial de *RoboCup*.

Os comités técnicos das diversas ligas têm a incumbência de definir o futuro de cada uma das ligas. Têm também como função iniciar e guiar a discussão referente à alteração das regras da respectiva liga, efectuada entre os membros da comunidade. Para além desta função, os comités técnicos são também responsáveis por tomar as decisões finais relativas às regras a utilizar, em cada liga, em cada ano. Embora inicialmente os comités técnicos das várias ligas fossem constituídos por membros designados pelo comité executivo, em 2001, a eleição dos membros dos comités foi democratizada. Desta forma, cada comité técnico passou a ser constituído por sete membros, quatro dos quais designados pelo comité executivo e três eleitos democraticamente pela comunidade de investigação na liga respectiva do *RoboCup*[45].

[45] O autor é desde Novembro de 2001, membro do comité técnico da Liga de Simulação do RoboCup. É o único elemento do comité técnico da liga de simulação que foi eleito em todas (duas) as eleições realizadas (2001 e 2002) para este órgão da federação RoboCup.

8.2 As Ligas do RoboCup

O desafio colocado pelo *RoboCup* aos investigadores mundiais das áreas da IA e Robótica subdivide-se em 3 categorias principais (figura 46), sendo que, o Futebol Robótico é claramente a categoria mais relevante:

- ***RoboCup Soccer* –** Futebol Robótico, incluindo uma liga de simulação e três ligas robóticas principais com regras distintas;
- ***RoboCup Rescue* –** Aplicação da investigação realizado no futebol robótico ao domínio da busca e salvamento em grandes catástrofes. A categoria divide-se numa modalidade de simulação e uma modalidade robótica;
- ***RoboCup Júnior* –** Utilização do *RoboCup* na educação. Crianças e jovens utilizando uma infra-estrutura simples, criam equipas de robôs para jogar futebol (2 contra 2), dançar ou resgatar vítimas.

Para além destas actividades e da organização anual do campeonato mundial de futebol robótico, a iniciativa RoboCup conta ainda com a realização de conferências especializadas, programas educacionais e desenvolvimento de infra-estruturas diversas.

Figura 46: As Três Vertentes do RoboCup

O Futebol robótico é sem dúvida o pilar central do *RoboCup*. Divide-se em quatro ligas principais e diversos desafios associados [RoboCup, 2001]:

- **Liga de Simulação –** Equipas de 11 agentes de *software* competem num ambiente simulado, jogando um desafio de futebol virtual, utilizando o sistema de simulação *soccerserver*;
- **Liga de Robôs Pequenos –** *Small-Size* – Equipas de 5 robôs de reduzidas dimensões, com controlo e visão centralizados, competem num campo com dimensões semelhantes às de uma mesa de *ping-pong*;
- **Liga de Robôs Médios –** *Middle-Size* – Equipas de 4 robôs totalmente autónomos competem num campo de 5x10 metros;
- **Liga de Robôs com Pernas –** *Sony Legged* – Equipas de 4 robôs cães Aibo da

Sony [Yamamoto e Fujita, 2000] jogam num campo de reduzidas dimensões.

Para além das ligas principais do *RoboCup*, existem diversos desafios associados:

- **Liga de Robôs Humanóides –** Desenvolvimento de robôs humanóides capazes de a médio prazo, jogar um desafio de futebol. Para já, têm existido unicamente demonstrações e competições de velocidade de deslocamento bípede e *penalties*.
- **Treinador da Simulação –** Desenvolvimento de sistemas de análise de jogo que sejam capazes de alterar o comportamento de uma equipa, através de instruções para os seus constituintes, tendo em consideração o decurso do jogo;
- **Comentador Inteligente –** Desenvolvimento de sistemas de análise de jogo e comentadores inteligentes com capacidade de geração de linguagem natural;
- **Visualizador 3D –** Criação de visualizadores tridimensionais para jogos de simulação, incluindo animações e sons realistas;

Todos os anos, de forma a aferir o progresso científico realizado pelos vários grupos de investigação, nos diversos tópicos mencionados, a Federação *RoboCup* organiza o *Robot World Cup Soccer Games and Conferences*, vulgarmente conhecido por *RoboCup*. Competições relacionadas com as diversas ligas e desafios, demonstrações e um congresso científico [Kitano, 1998] [Asada e Kitano, 1999] [Veloso et al., 2000] [Stone et al., 2001a] [Birk et al., 2002] (descrito no anexo 3) compõem o programa deste evento. Existem ainda diversas competições e eventos regionais relacionados com o *RoboCup* tais como o campeonato Europeu, o *German Open*, o *Japan Open*, o *China Open,* etc.[46].

As equipas portuguesas têm obtido muito bons resultados no âmbito do RoboCup, destacam-se a vitória num campeonato do Mundo e dois campeonatos da Europa da liga de simulação por parte da equipa FC Portugal, a vitória num campeonato, a vitória num campeonato da Europa da liga de robôs pequenos por parte da equipa 5dpo [Costa et al., 2000] e dois terceiros lugares em campeonatos da Europa da liga de robôs médios por parte da equipa 5dpo [Moreira et al., 2002].

8.2.1 A Liga de Simulação

A liga de simulação é baseada no sistema de simulação de utilização livre *soccerserver* [Noda et al., 1998] [SoccerServer, 2001]. Este sistema simula um campo contendo 2 equipas e uma bola onde se realiza um jogo 2D de futebol. Cada equipa é composta por 11 jogadores (e eventualmente 1 treinador) que se conectam ao simulador, numa arquitectura cliente-servidor, através de *sockets* UDP. O simulador aceita comandos de baixo-nível dos

[46] Em 2003 surgiram diversas novas competições regionais incluindo o Australian Open, American Open, Iranian Open e o festival Robótica 2003 em Portugal cujo tema principal foi o futebol robótico.

jogadores, executa-os de forma imperfeita e envia informação (também imperfeita) de percepção aos jogadores.

O *soccerserver* possibilita a realização de investigação de alto-nível em coordenação, aprendizagem e planeamento multi-agente, enquanto se espera pelo desenvolvimento de hardware suficientemente robusto e apropriado, que permita efectuar este tipo de investigação com robôs reais. Os avanços na liga de simulação incluem a utilização de aprendizagem para optimizar as habilidades de baixo-nível [Stone, 1998] [Riedmiller et al., 2000], técnicas de cooperação tais como tácticas [Reis e Lau, 2003a], formações [Reis, et al., 2001] [Stone e Veloso, 1999] e papéis [Reis et al., 2001] [Stone, 1999], formalizações de estratégias de equipa complexas [Prokopenko e Buttler, 1999] [Reis et al., 2001] [Reis e Lau, 2003a], planeamento flexível [Stone, 1999], sistemas de comunicação inteligente [Stone, 1999] [Prokopenko e Buttler, 1999] [Reis e Lau 2001a, 2001], novas arquitecturas de agentes [Stone, 1998] [Reis e Lau 2001a, 2001], linguagens de comunicação para agente e para treinadores [Reis e Lau, 2002] [SoccerServer, 2001] e ferramentas de *debug* e análise de agentes [Stone, 1999] [Reis et al., 2001].

A aprendizagem tem sido um dos temas mais proeminentes de investigação no âmbito da liga de simulação [Stone et al., 2001c]. Grande parte das melhores equipas na liga de simulação tais como o CMUnited [Stone, 2000a] e os Karlsruhe Brainstormers [Riedmiller et al., 2000] têm desenvolvido os seus *skills* básicos (e até comportamentos individuais e cooperativos) com o auxílio de aprendizagem por reforço [Stone, 1998] [Riedmiller et al., 2000]. No entanto, outras equipas bem sucedidas têm demonstrado a possibilidade de desenvolver muito bons comportamentos de baixo-nível sem o recurso a aprendizagem. Exemplos disto são a equipa FC Portugal que utiliza optimização *on-line* nos seus *skills*, Tsinghuaeolus [Cai et al., 2002] e YowAI que utilizam abordagens analíticas no desenvolvimento destes *skills*.

A investigação em estratégias de equipa complexas e em mecanismos de coordenação para Sistemas Multi-Agente tem sido outra das áreas de maior destaque na liga de simulação, ao longo dos últimos anos. Técnicas de coordenação como formações e papéis foram introduzidas por Stone e Veloso [Stone, 1998] [Stone e Veloso, 1999] no *RoboCup* e conduziram à criação da equipa CMUnited que foi campeã do mundo em 1998 e 1999. Reis, Lau e Oliveira estenderam o trabalho de Stone e Veloso e criaram protocolos flexíveis para efectuar a troca de papéis e posicionamentos de agentes heterogéneos [Reis et al., 2001], introduziram o conceito de táctica no *RoboCup* [Reis et al., 2001] e criaram um mecanismo de posicionamento que permite a definição de formações muito semelhantes às utilizadas no futebol real [Reis et al., 2001]. Baseada nestes conceitos foi desenvolvida a equipa FC Portugal que foi campeã do mundo do RoboCup em 2000 [Reis e Lau, 2001a]. Este trabalho foi ainda estendido de forma a criar uma formalização genérica para o conceito de estratégia para uma competição contra um oponente com objectivos contrários [Reis et al., 2001] [Reis e Lau, 2003a]. Este modelo, aplicável também a outros domínios, constitui uma das bases para o desenvolvimento da

flexibilidade da equipa FC Portugal e para o seu sucesso no *RoboCup* [Lau e Reis, 2000] [Lau e Reis, 2002] [Reis e Lau, 2003a].

A liga de simulação, o funcionamento do sistema de simulação e a investigação realizada no âmbito desta liga vai ser objecto de uma análise mais profunda ao longo das diversas secções que compõem este capítulo. A investigação realizada no âmbito do projecto FC Portugal na liga de simulação será objecto de análise no capítulo 9.

8.2.2 A Liga de Robôs Pequenos – *Small-Size*

A liga de robôs pequenos (*small-size*) é também conhecida como a liga F-180. Cada jogo disputa-se entre equipas compostas por 5 robôs de reduzidas dimensões, que jogam num campo com alcatifa verde e marcações brancas com dimensões ligeiramente superiores às de uma mesa de *ping-pong*. As regras do jogo permitem a utilização de visão global e controlo centralizado dos robôs. Desta forma, durante o jogo, o sistema de controlo dos robôs, geralmente externo a estes, recebe a informação proveniente da câmara localizada sobre o campo, processa esta informação determinando as posições, orientações e velocidades dos robôs e da bola, decide o comando mais apropriado a ser executado por cada robô e envia este comando através de comunicação rádio a cada um dos robôs. Embora a maioria das equipas utilize visão e controlo centralizado, algumas equipas têm utilizado visão e decisão locais, o que embora dificulte a sua capacidade para jogar, torna os problemas de investigação relacionados com a visão e localização muito mais ricos.

Figura 47: A liga de Robôs Pequenos (Small-Size)

A superfície de jogo é forrada com alcatifa verde delimitada por paredes brancas. Uma das balizas está pintada de azul e a outra de amarelo. Os robôs possuem marcas próprias no topo de forma a poderem ser identificados pelo sistema de visão global. Os robôs da liga de *small-size* podem ter até 18 cm segundo qualquer diagonal e ocupar, no máximo, 180

cm^2. Cada equipa pode ter um máximo de 5 robôs que, no entanto, podem apresentar características distintas. Cada jogo desta liga consiste em duas partes de 10 minutos cada. Durante o jogo nenhuma interferência humana é permitida com o sistema de controlo dos robôs.

A necessidade de alta velocidade e de um controlo preciso deram a esta liga a reputação de "a liga de engenharia" [Stone et al., 2001b]. Os desenvolvimentos científicos realizados no âmbito desta liga incluem [Stone et al., 2001b] novos sistemas electromecânicos, de controlo de sistemas, de electrónica digital e comunicações sem fios. A maioria das equipas tem investido os seus esforços de investigação nestes tópicos e unicamente algumas equipas, até 2000 (tais como os campeões em 1999 e 2000 - Cornell BigRed), demonstraram comportamentos de equipa tais como passes explícitos, papéis ou formações. No entanto, os avanços verificados ao longo dos últimos anos no hardware e a relativa estabilidade das regras do jogo contribuem para uma necessidade urgente da realização de investigação em metodologias de coordenação para ser bem sucedido nesta liga, ao longo dos próximos anos [Reis, 2001a].

8.2.3 A Liga de Robôs Médios – *Middle-Size*

A liga de robôs médios (*middle-size*) é também conhecida como a liga F-2000 e coloca um vasto conjunto de problemas de investigação que despertou o interesse de mais de 50 grupos de investigação espalhados pelo mundo. Grande parte dos problemas de investigação colocados na liga de simulação, tais como autonomia dos agentes, visão local e na liga de robôs pequenos, tais como processamento de imagem complexo, necessidade de controlo preciso em tempo-real, encontra-se simultaneamente na liga de robôs médios.

Três factores influenciam primariamente o projecto das equipas e dos seus robôs:

- O ambiente de jogo, nomeadamente o campo;
- As restrições impostas ao projecto dos robôs;
- A tarefa cooperativa a realizar, isto é, jogar futebol.

O ambiente de jogo foi projectado cuidadosamente para que os problemas relacionados com a percepção e locomoção sejam suficientemente simples mas simultaneamente constituam desafios estimulantes, que permitam despertar investigação séria no seio desta liga [Veloso et al., 2000] [RoboCup, 2001]. O campo é forrado a alcatifa verde e possui 10x5m de dimensão e duas balizas coloridas respectivamente de amarelo e azul. Até 2001 o campo estava rodeado de paredes brancas (com 50 cm de altura). Em 2002 as regras foram alteradas, retirando as paredes que rodeavam o campo e colocando postes coloridos nos quatro cantos, de forma a permitir uma localização mais simples aos robôs. Diversas linhas brancas são utilizadas para marcar o campo, incluindo uma linha que delimita o campo, linhas de golo, linhas que marcam as áreas, uma linha central e uma linha que delimita o círculo central. A iluminação do campo deve estar compreendida entre 500 e

1500 lux. De forma a serem regulamentares os robôs devem possuir um corpo negro e possuir marcas de identificação da equipa (*magenta* ou *cyan*). Diversas restrições elaboradas às dimensões, peso e forma dos robôs são colocadas de forma a garantir alguma uniformidade nas características dos mesmos.

Figura 48: A liga de Robôs Médios (Middle-Size)

Os jogos de robôs médios são disputados entre duas equipas compostas por quatro robôs cada. Um árbitro humano e um assistente encarregam-se de fazer cumprir as regras do jogo. Cada jogo é composto por duas partes com 10 minutos cada.

8.2.4 A Liga de Robôs com Pernas – Cães da Sony

A liga de robôs com pernas utiliza equipas de 4 robôs AIBO ERS210 construídos pela Sony. Os robôs AIBO têm 20 graus de liberdade e sete tipos de sensores: imagem, áudio, temperatura, proximidade, aceleração, pressão e vibração. Nesta liga, uma vez que não é possível efectuar modificações no hardware dos robôs, a equipa com o *software* mais apropriado normalmente vence os jogos.

Cada jogo na liga de robôs com pernas é composto por duas metades de 10 minutos cada. Os robôs têm de ser totalmente autónomos durante os jogos. Controlo remoto, comunicação sem fios, ou visão global não foram permitidos até 2001 (embora comunicação visual e sonora pudessem ser utilizadas). A partir de 2002 foi permitida a realização de comunicação sem fios entre os robôs dentro do campo (embora este tipo de comunicação permanecesse proibida com o exterior do campo)

Os robôs jogam num campo de reduzidas dimensões (aproximadamente 4x3 metros) que contém marcadores bem definidos com determinadas cores. O campo tem também um rebordo branco que evita que a bola saia do mesmo. A bola é laranja e as balizas são azuis e amarelas. Os marcadores coloridos são utilizados para auxiliar os robôs a localizarem-se.

Nove cores são utilizadas no campo: cor de rosa e verde (para os marcadores), amarelo e azul claro (para os marcadores e balizas), vermelho e azul escuro (para os equipamentos dos robôs), cor de laranja (para a bola), verde claro (para o campo) e branco (para as linhas e outras marcações no campo). Os robôs necessitam de ter capacidade para detectar e discriminar estas cores de forma a serem capazes de reconhecer os objectos presentes no campo.

A investigação realizada no âmbito da liga de robôs com pernas está actualmente centrada em visão por computador, localização e locomoção robótica. Para estes três desafios de investigação, diferentes grupos desenvolveram diferentes abordagens. Com a recente disponibilização de comunicação sem fios e o aumento do número de elementos de cada equipa de 3 para 4, a capacidade de coordenação e comportamento de equipa dos robôs passou a ser crucial a partir de 2002.

A visão é um problema muito complexo nesta liga. Na liga de robôs pequenos a visão é obtida por um sistema centralizado através de uma (ou mais) câmaras colocadas em cima do campo. Através deste sistema é relativamente simples identificar as posições da bola e de todos os robôs que se encontram sobre o campo. Na liga de robôs médios a visão é bastante mais complexa pois os robôs têm de ser totalmente autónomos e como tal carregar os seus sistemas de visão totalmente a bordo. Este facto provoca que cada robô só tenha uma visão parcial do campo. Acresce que, habitualmente robôs de diferentes tamanhos e formas estão presentes no campo, parcialmente ocultos, o que torna o reconhecimento de objectos um problema complexo.

Figura 49: A liga de Robôs com Pernas da Sony

Na liga de robôs com pernas da Sony, a visão é ainda um problema mais complexo. Os robôs têm um ângulo de visão muito limitado e devido à locomoção com pernas, um acentuado efeito de oscilação está presente nas imagens. A cabeça dos robôs onde a câmara está montada possui três graus de liberdade o que permite a implementação de

algoritmos de percepção visual inteligente. Embora a Sony disponibilize, em conjunto com a plataforma robótica, algoritmos de detecção de cores e processamento de imagem, a maioria das equipas desenvolve os seus próprios algoritmos de processamento e análise de imagem. Por exemplo, Bruce et al. [Bruce et al., 2000] desenvolveu uma biblioteca de processamento e análise de imagem disponível utilizando a licença GPL designada CMVision [Bruce et al., 2001].

A localização é também um problema extremamente complexo na liga de robôs com pernas. De modo a permitir aos robôs efectuarem decisões baseadas em objectivos, a posição e direcção dos robôs no campo é extremamente importante [Wendler, 2001]. Os robôs têm três graus de liberdade para movimentar a câmara e, deste modo, é possível desenvolver algoritmos de localização baseados no movimento da cabeça de forma a localizar os marcadores posicionados em torno do campo que permitem aos robôs localizarem-se. A qualidade das imagens obtidas é crucial para definir algoritmos de localização robustos. É necessário desenvolver algoritmos de processamento e análise de imagem que permitam não só reconhecer os objectos presentes na imagem, mas também medir com precisão as dimensões, distâncias e direcções dos objectos presentes na imagem. A informação visual disponível neste ambiente, extremamente dinâmico, contém grandes erros, incluindo potenciais erros catastróficos (tais como a detecção de objectos fantasmas). Diferentes equipas desenvolveram diferentes algoritmos para a localização tais como métodos probabilísticos ou raciocínio baseado em casos[47].

Embora algoritmos para efectuar a movimentação básica dos robôs sejam disponibilizados pela Sony em conjunto com os robôs AIBO, estes não são suficientemente poderosos para desenvolver uma equipa competitiva nesta liga. As equipas têm de desenvolver melhores algoritmos de actuação a baixo-nível, incluindo algoritmos para andar rápido, rodar rápido, driblar, chutar ou defender a bola. Para além disso, a selecção de comportamentos de alto-nível é também muito importante. Por exemplo, os campeões de 2000 e 2001, UNSW da University de New South Wales em Sidney, Austrália, utilizaram robôs com diferentes papéis, cada qual com uma selecção de comportamentos distinta.

Outros tópicos de investigação relevantes nesta liga incluem a definição de arquitecturas de agentes de *software* apropriadas, comunicação multi-agente (*wireless* ou por áudio) entre robôs, e algoritmos para comportamento cooperativo em equipa. Embora todas as equipas utilizem um conjunto de módulos de *software* típico na sua arquitectura (tais como a visão, localização e locomoção), algumas equipas optaram por arquitecturas de agente distintas, utilizando, por exemplo, princípios BDI.

[47] Case-based reasoning

8.2.5 O *RoboCup Rescue:* Salvamento em Catástrofes

O salvamento em situações de desastre é um dos problemas sociais mais relevantes da actualidade, envolvendo um elevado número de agentes heterogéneos que actuam num ambiente hostil. O projecto *RoboCup Rescue* foi lançado em 1999 com a intenção de promover investigação e desenvolvimento neste domínio de grande impacto social a diversos níveis, envolvendo a coordenação do trabalho de equipas de agentes múltiplos, agentes físicos para busca e salvamento, infra-estruturas de informação, assistentes digitais pessoais e um simulador realista. Utilizando estes sistemas pretende-se efectuar a avaliação de estratégias e sistemas robóticos para busca e salvamento, sendo futuramente todos estes subsistemas integrados de uma forma abrangente.

Este projecto assenta na experiência de sucesso do projecto RoboCup na sua vertente de futebol de forma a permitir aplicar as metodologias desenvolvidas neste domínio em outras áreas, socialmente mais úteis. O domínio foi escolhido, em parte também, pelas suas características semelhantes ao futebol, como sejam o ambiente dinâmico, informação incompleta e com ruído, agentes heterogéneos e necessidade de coordenação de equipas de agentes na execução de tarefas complexas. A similaridade dos domínios leva os investigadores da área a crer que as metodologias definidas para as ligas de futebol do RoboCup, poderão em grande parte ser aplicadas também no *RoboCup Rescue* [Kitano et al., 1999].

Os objectivos principais do projecto podem ser detalhados da seguinte forma [Stone et al., 2001c]:

- Desenvolvimento e aplicação de tecnologias avançadas de robótica inteligente e inteligência artificial, para resposta a emergências e desastres no âmbito de sistemas sociais mais seguros;
- Introdução de novos problemas de importância social, como desafios para a robótica e IA, indicando direcções de investigação interessantes;
- Proposta de infra-estruturas de sistemas futuros baseados essencialmente em robótica e IA;
- Aceleração da investigação e desenvolvimento em busca e salvamento através da introdução da competição RoboCup.

Dois projectos e respectivas ligas com competições anuais estão actualmente activos: *Projecto de Simulação, Projecto de Robótica e Infra-estrutura*. Destes dois projectos resultam duas ligas com competições anuais: *Liga de Simulação de Rescue* e *Liga de Robôs de Rescue*. Os participantes dos dois projectos afirmam que a integração destas actividades conduzirá à criação das equipas de agentes robóticos heterogéneos de busca e salvamento do futuro [Rescue, 2001].

O *Projecto de Simulação RoboCup Rescue* iniciou-se com a construção de um ambiente genérico de simulação de desastres urbanos baseado em redes de computadores e na tecnologia dos agentes inteligentes. À semelhança do simulador *soccerserver*, o simulador de *Rescue*, inclui um módulo de simulação e um módulo de visualização. Diversos visualizadores foram construídos, incluindo visualizadores tridimensionais (figura 50).

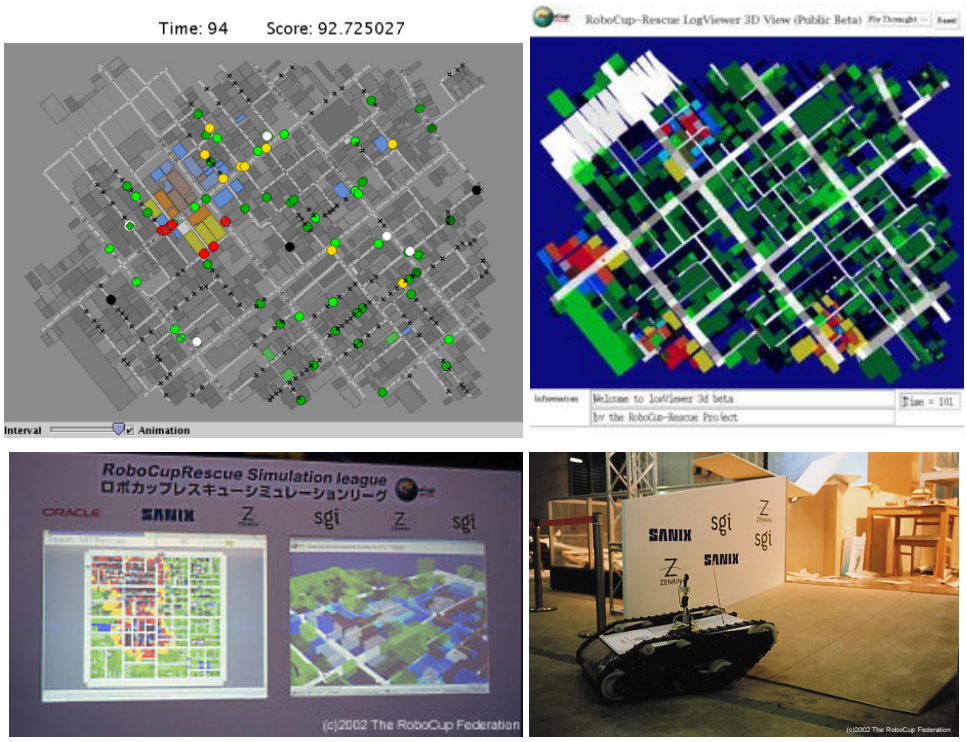

Figura 50: RoboCup Rescue (Visualizadores – em cima e Competição – em baixo)

O simulador modeliza um mundo virtual (cidade virtual) e a ocorrência de uma catástrofe nesse mundo. Para tal, simula o colapso de edifícios, bloqueio de estradas, espalhamento de incêndios, fluxo de trânsito e os seus efeitos mútuos. Diversos agentes heterogéneos inteligentes, tais como bombeiros, polícias, vítimas, voluntários, etc., conduzem acções de busca e salvamento neste mundo virtual. O objectivo das equipas de agentes consiste em minimizar os desastres da catástrofe e evitar a perda de vidas de humanas.

8.2.6 O RoboCup Júnior

O *RoboCup Júnior* é uma iniciativa onde as crianças do ensino básico e secundário podem experimentar na prática tópicos avançados de robótica. É essencialmente um projecto educacional com o objectivo de providenciar um ambiente que permita às crianças aprender ciência e tecnologia em geral através da robótica [Kitano, 1999] [Sklar et al., 2002]. Devido ao objectivo emblemático do RoboCup (que em 2050 uma equipa de robôs vença os campeões humanos de futebol), torna-se necessário que as gerações mais jovens se envolvam e como tal o *RoboCup Jr.* surge nesse contexto.

O *RoboCup Jr.* foi projectado de forma a que a excitação do jogo de futebol utilizando robôs reais em vez de criaturas virtuais possa estimular os mais novos, dando-lhes a conhecer uma parte da complexidade que é realmente programar robôs para desempenharem tarefas complexas. Ao contrário das outras ligas, que foram projectadas para permitir investigação ao mais alto nível por parte dos laboratórios de investigação, universidades e empresas, o *RoboCup Júnior* procura utilizar hardware robótico genérico e de fácil programação, tal como os MindStorms da Lego, para permitir a crianças terem uma experiência totalmente prática de realmente construir e programar um robô.

Figura 51: RoboCup Júnior (Modalidades de Futebol e Dança)

O *RoboCup Júnior* oferece diversos desafios com ênfase em aspectos cooperativos e competitivos. Para as crianças, esta iniciativa providencia uma oportunidade para trabalharem em grupo enquanto são introduzidos aos princípios básicos da robótica, electrónica, *hardware* e *software*. Os desafios actualmente oferecidos (figura 51) são três [RoboCup, 2001]:

- **Futebol.** Equipas de 2 robôs autónomos competem entre si num campo rectangular de reduzidas dimensões (figura 51).
- **Dança.** Um ou mais robôs executam habilidade de dança ao som da música. Este desafio procura estimular a criatividade na construção dos robôs e na definição dos seus movimentos.
- **Salvamento.** Os robôs executam corridas para salvar vítimas de um suposto desastre. O campo varia em complexidade, desde o simples seguimento de uma linha numa superfície plana, até ao seguimento de um caminho com obstáculos e inclinações.

Actualmente encontra-se em fase de construção um simulador oficial para o *RoboCup Júnior*, com o objectivo de permitir envolver ainda mais estudantes pré-universitários nesta iniciativa.

8.2.7 Outros Desafios Associados

8.2.7.1 Treinador *On-Line*

O treinador on-line é um agente especial na liga de simulação que possui informação global e sem erros sobre o jogo realizado. Como não está sujeito às restrições relacionadas com o raciocínio em tempo-real como estão os agentes que controlam os diversos jogadores (o treinador não controla qualquer jogador), este agente é utilizado para efectuar análises mais complexas, como as análises estratégicas ou tácticas do jogo.

Um dos desafios de investigação relacionados com o treinador *on-line* consiste na definição de uma linguagem de comunicação que permita a este agente comunicar de forma simples com a sua equipa de forma a alterar a táctica de jogo de acordo com a sua análise estratégica. Reis e Lau desenvolveram a primeira linguagem de alto-nível para o treinador, COACH UNILANG [Reis e Lau, 2002] no âmbito da equipa FC Portugal. Baseada nesta linguagem e nos contributos de diversos investigadores foi desenvolvida uma linguagem mais simples e menos expressiva, designada CLang, com o objectivo de permitir modificar o comportamento individual de agentes da liga de simulação e efectuar uma competição entre treinadores. Esta linguagem foi incorporada no *soccerserver* e designada "linguagem standard". No entanto, unicamente 4 equipas participaram no RoboCup 2001 na primeira competição de treinadores que a linguagem deveria permitir realizar, tendo as outras equipas concluído que a linguagem era demasiado "baixo-nível" e preocupado com o comportamento individual para permitir efectuar alterações no comportamento global da equipa. Em 2002 diversos conceitos da linguagem COACH UNILANG foram introduzidos na linguagem CLang tornando-a bastante mais expressiva e adequada à realização de uma competição de treinadores.

8.2.7.2 Visualizadores 3D

O desenvolvimento de visualizadores 3D para as diversas competições RoboCup tem sido um desafio de investigação estimulado ao longo dos últimos anos. O domínio preferencial da aplicação dos visualizadores tem sido a liga de simulação de futebol robótico. No entanto, nos últimos anos, após o aparecimento do *RoboCup Rescue*, visualizadores 3D para este domínio de simulação têm também sido desenvolvidos. Uma análise mais detalhada dos sistemas de visualização 3D disponíveis, dos projectos de investigação realizados nesta área, e do visualizador 3D desenvolvido no âmbito do projecto FC Portugal será apresentada na secção 9.9.

8.2.7.3 Comentador Inteligente

O desenvolvimento de um sistema de análise do jogo e comentário inteligente em linguagem natural, é um dos desafios associados ao RoboCup. O sistema tem de ser capaz de efectuar tarefas idênticas às efectuadas pelas equipas tais como detectar a posição da bola e dos jogadores de cada equipa no campo. No entanto, adicionalmente, tem de ser capaz de calcular informação de mais alto-nível a partir da evolução temporal das posições dos jogadores e da bola. Esta informação pode consistir em informação estatística do jogo, informação relativa ao comportamento colectivo de cada uma das equipas e informação relativa ao comportamento individual de cada jogador. A informação estatística relevante inclui[48] o resultado (número de golos marcados por cada equipa), o tempo de posse de bola de cada equipa em cada região do campo, o número de remates efectuados por cada equipa, informação relativa às perdas de bola e recuperações de bola, passes correctos e errados de cada equipa em cada região, etc. Esta análise pode implicar um sistema de visão centralizado, no caso do comentador ser aplicado às ligas robóticas (*small-size, middle-size* ou *legged*).

Para além da análise ao jogo, o comentador deve ser capaz de interpretar a informação e gerar comentários em linguagem natural baseados na análise efectuada e nos acontecimentos que se estão a passar no jogo. Isto implica a realização de investigação em tópicos tais como o processamento de linguagem natural e síntese de voz.

Embora este desafio seja amplamente encorajado pela Federação RoboCup (essencialmente devido à atracção que propicia para os espectadores), poucos sistemas de análise e comentário foram desenvolvidos até ao momento. De entre os sistemas desenvolvidos destacam-se o MIKE e o Isaac.

O sistema Isaac foi desenvolvido por Tambe et al. [Tambe et al., 1999] com o objectivo de efectuar análises à posteriori de jogos efectuados na liga de simulação. O sistema utiliza o algoritmo C4.5 para gerar árvores de decisão que demonstram como as equipas se comportaram num dado jogo. No entanto as análises que realiza são em geral muito simples e longe das análises realizadas por um analista de futebol real.

O sistema MIKE foi desenvolvido originalmente em 1998 pelo laboratório ETL no Japão para a liga de simulação [Frank, 2001]. É capaz de gerar comentários em texto ou utilizando síntese de voz em Inglês, Francês ou Japonês. Em conjunto com o sistema foi desenvolvido um sistema de cálculo de estatísticas de jogo, o *Proxy Server* capaz de se conectar ao simulador durante o jogo e gerar diversas estatísticas simples de jogos realizados na liga de simulação.

---

[48] Uma análise mais detalhada da informação estatística relevante na liga de simulação será efectuada no capítulo 9, durante a análise da linguagem de treinador COACH UNILANG.

O sistema MIKE foi entretanto adaptado com sucesso para a liga de *small-size*. Esta adaptação foi efectuada utilizando um sistema de visão global que envia a posição dos robôs e da bola para o sistema MIKE e na sua associação ao robô humanóide SIG que se encarregava posteriormente de realizar algumas animações e efectuar o comentário [Frank, 2001].

8.3 O Futebol Robótico Simulado

A liga de simulação do RoboCup é baseada no simulador designado por *SoccerServer* [Noda et al., 1998] que simula um jogo virtual de futebol disputado entre duas equipas compostas por 11 agentes autónomos cada. O simulador foi construído como um ambiente de simulação multi-agente, incerto e com funcionamento em tempo-real, capaz de permitir a competição entre equipas de jogadores virtuais, cada qual controlado separadamente por um agente autónomo.

Esta secção pretende oferecer uma panorâmica geral do funcionamento do simulador que é utilizado como base para as competições de futebol robótico simulado e para diversos outros trabalhos de investigação e desafios associados – o *soccerserver*. O conteúdo da secção é baseado no manual da versão actual do simulador – *soccerserver* [SoccerServer, 2001] e na experiência de utilização do sistema do autor. São apresentadas essencialmente as características mais relevantes do simulador para a compreensão do trabalho realizado no âmbito desta tese.

8.3.1 Aplicações Constituintes do Simulador

O sistema de simulação de futebol é composto por 3 módulos principais: o servidor (*soccerserver*); o visualizador (*soccer monitor*) e o vídeo (*logplayer*).

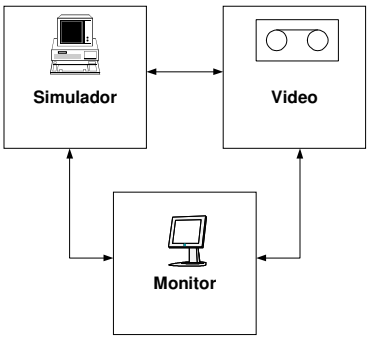

Figura 52: Aplicações Constituintes do Sistema de Simulação

O simulador é o responsável por realizar a simulação de jogos utilizando para tal uma arquitectura cliente-servidor. Providencia um campo de futebol virtual e simula o movimento de todos os objectos no mesmo, controlando o jogo segundo um conjunto de regras pré-especificado. Os clientes (jogadores) conectam-se ao simulador através de

sockets UDP. O simulador recebe os comandos dos clientes, executa-os simulando o movimento de todos os objectos no campo e envia aos clientes informação sensorial.

O monitor é a ferramenta que permite visualizar os jogos virtuais realizados, com o auxílio do simulador. Comunica com o simulador também através de *sockets UDP,* recebendo deste as posições de todos os objectos no campo virtual. Efectua depois o *display* gráfico desta informação de forma a permitir aos humanos a visualização do jogo.

O vídeo (*Logplayer*) é uma aplicação que permite a visualização de jogos pré-gravados (ficheiros log) à semelhança do que acontece com um gravador de vídeo. A sua utilização é associada à utilização de um monitor de forma a poder visualizar os jogos.

8.3.1.1 O Simulador – *SoccerServer*

O simulador (servidor) é o sistema que possibilita a realização de jogos simulados de futebol entre duas equipas compostas por 11 agentes autónomos. Uma vez que funciona numa arquitectura cliente-servidor, permitindo aos jogadores comunicarem através de *sockets UDP*, não impõe qualquer limitação às linguagens ou sistemas operativos utilizados para o desenvolvimento das equipas. A única restrição é que as equipas suportem comunicação via *UDP/IP*.

Cada equipa pode ter até 11 jogadores (clientes) e eventualmente um treinador (cliente com capacidades e privilégios especiais). Cada cliente é um processo separado que se conecta ao servidor através de uma porta especificada. Para participar num jogo, o cliente deve enviar uma mensagem para a porta 6000 do servidor que como resposta deverá aceitar a sua conexão atribuindo-lhe uma porta específica que irá ser utilizada a partir desse momento para efectuar a transferência de informação entre o simulador e o cliente.

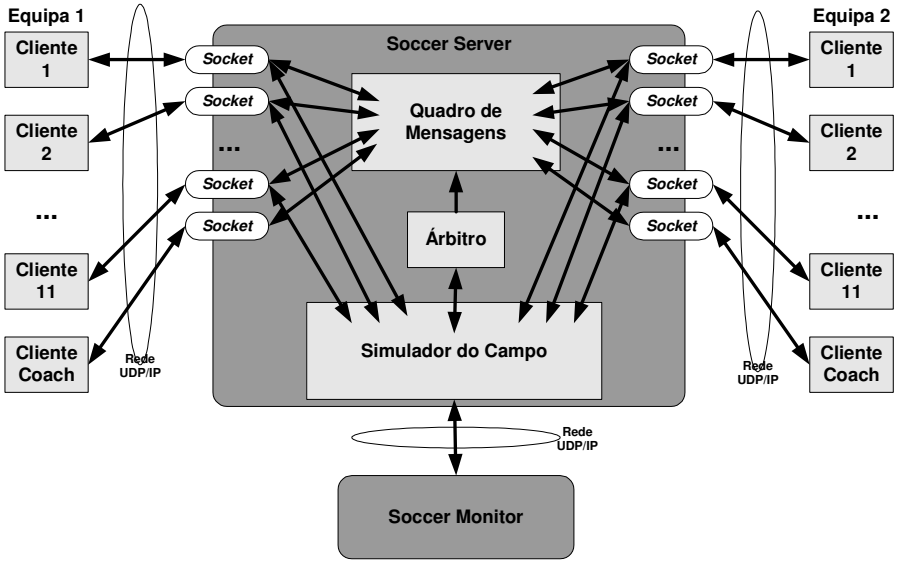

Figura 53: Arquitectura do Sistema de Simulação SoccerServer

Os jogadores enviam pedidos ao simulador sobre as acções a realizar pelo jogador que controlam (rodar, acelerar, chutar a bola, etc.). O servidor recebe as suas mensagens, trata de executar os seus pedidos (deste que estes sejam válidos[49]) e actualiza o ambiente em consonância. Em adição, o servidor envia aos jogadores informação sensorial, i.e., informação visual relativa aos jogadores e objectos no campo de visão do jogador, informação auditiva composta pelas mensagens que o jogador ouve, etc.

O simulador é um sistema de tempo real que no entanto trabalha com intervalos de tempo discretos, ou ciclos. Em cada ciclo (que tem uma duração por defeito de 100ms), o simulador aguarda por comandos dos jogadores e envia (se for o caso), informação sensorial aos jogadores. No final do ciclo, o simulador actualiza o ambiente virtual tendo em conta as acções que cada jogador deseja executar e a dinâmica do sistema em geral.

8.3.1.2 O Monitor – *Soccer Monitor*

O *soccer monitor* é uma ferramenta de visualização que permite visualizar os jogos realizados utilizando o simulador *soccerserver*. Recebe do simulador, através de *sockets UDP*, a informação respeitante aos diversos objectos e suas posições no campo e aos eventos do jogo. A visualização desta informação é realizada através de uma interface gráfica. Permite ainda comunicar em sentido inverso com o simulador (servidor), permitindo desta forma ao utilizador, enviar comandos úteis como sejam o início ou interrupção de um jogo ou a marcação de um livre numa dada região do campo a favor de uma das equipas.

Vários monitores podem ser conectados ao mesmo servidor no decurso de um jogo. Esta possibilidade permite visualizar em diversos terminais o mesmo jogo ao mesmo tempo utilizando para tal, diferentes ferramentas de visualização com potencialidades distintas (por exemplo um monitor 3D e um sistema de análise de jogo). No entanto, para realizar um jogo não é necessário conectar qualquer monitor ao simulador.

Monitores com capacidade de conexão ao simulador e de visualização gráfica dos jogos foram construídos por diferentes grupos de investigação. De entre estes, destacam-se os monitores tridimensionais que permitem uma visualização 3D dos jogos realizados e visualizadores direccionados para a análise de jogos e cálculo de estatísticas de jogo.

O monitor tradicional do *soccerserver* permite configurar as cores e dimensões com que são visualizados todos os objectos (jogadores e bola). Permite ainda configurar os tipos de mensagens visualizados e os respectivos textos e as dimensões e fontes destes. Embora desde a criação da liga de simulação, o monitor utilizado em jogos oficiais tenha sido o

[49] Exemplos de pedidos inválidos são: chutar a bola quando ela não se encontra suficientemente próxima do agente; acelerar o agente quando este não possui energia para o fazer; agarrar a bola quando o agente não é o guarda-redes da equipa.

Soccer Monitor tradicional, o monitor oficial do futebol simulado é actualmente, desde Outubro de 2001, o *FrameView* [Merke, 2002], construído pela universidade de Karlsruhe na Alemanha. Para além das funções do monitor tradicional, permite ainda analisar diversas características do jogo, úteis para a sua análise. Entre estas incluem-se a visualização de: ampliações de regiões do campo, da energia dos jogadores, da visão dos jogadores e das características dos jogadores heterogéneos[50] presentes no campo.

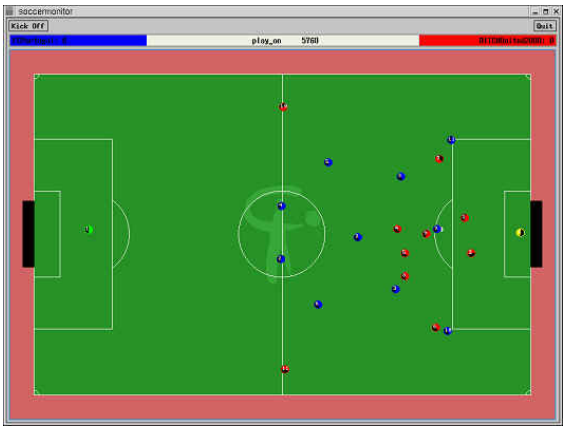

Figura 54: Monitor Tradicional para Linux do Soccer Server.

A arquitectura modular do simulador permite a separação entre o módulo de simulação e o módulo de visualização. Isto permite não só a utilização de diferentes monitores associados ao mesmo simulador, mas ainda a construção de monitores em sistemas operativos distintos. Um exemplo significativo é o visualizador para Windows construído por Klaus Dorer na universidade de Freiburg [Dorer, 2000] e continuado pela universidade de Tsinghua, China [Cai et al., 2002].

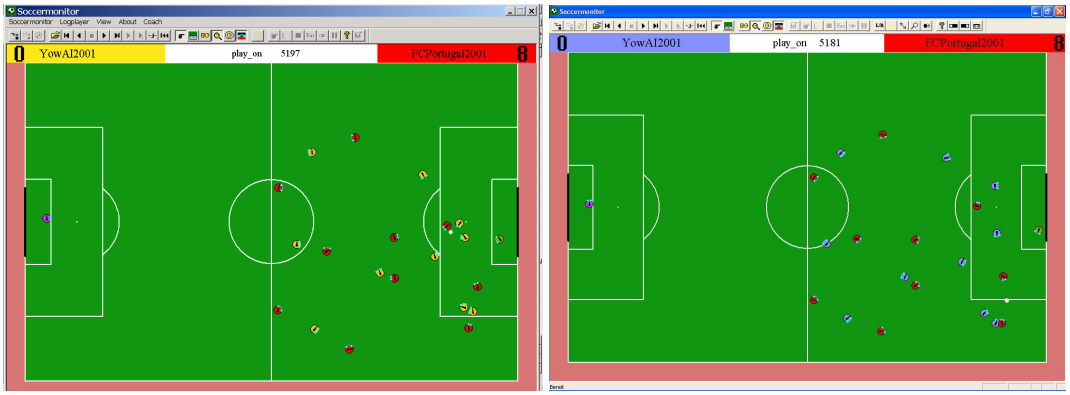

Figura 55: Monitores para Windows do Soccer Server (Freiburg e Tsinghua).

[50] Os jogadores heterogéneos foram introduzidos na liga de simulação em 2001. Consistem em jogadores com capacidades físicas distintas dos jogadores tradicionais. Por exemplo, um jogador pode ter uma maior velocidade máxima mas cansar-se mais rapidamente ou então ter uma maior potência de chuto mas uma menor precisão.

8.3.1.3 O Vídeo - *LogPlayer*

O *LogPlayer* é uma aplicação que permite a visualização de jogos pré-gravados à semelhança do que acontece com um gravador de vídeo. O servidor está equipado com opções que permitem, enquanto um jogo é realizado, gravar a informação mais relevante do mesmo para o disco do computador. O ficheiro gerado (ficheiro log) pode depois ser utilizado para visualizar o jogo com o auxílio do *Logplayer*. A utilização do *Logplayer* é usualmente combinada com a utilização de um Monitor de forma a visualizar o jogo gravado. A utilização principal do *LogPlayer* situa-se ao nível da análise da estratégia das várias equipas e dos seus pontos fracos e fortes. À semelhança do que se passa com um gravador de vídeo, o *LogPlayer* está equipado com botões de avanço e recuo rápido e paragem, dispondo também da possibilidade de avançar directamente para um dado ciclo do jogo.

8.3.2 O Mundo Simulado e as Regras do Jogo

Nesta secção descreve-se a forma de execução de um jogo de futebol robótico simulado, incluindo as regras do jogo e a dinâmica do mundo simulado. Para uma melhor compreensão do conteúdo desta e das seguintes secções, é necessário possuir conhecimentos básicos sobre futebol real e sobre as regras deste jogo.

8.3.2.1 O Campo e os Objectos

Os jogos de futebol robótico simulado disputam-se num campo virtual com 105*68m que contém diversas linhas e objectos para auxiliar os jogadores a se localizarem. A figura 56 contém uma representação do campo virtual e das linhas e marcos de localização que o mesmo contém.

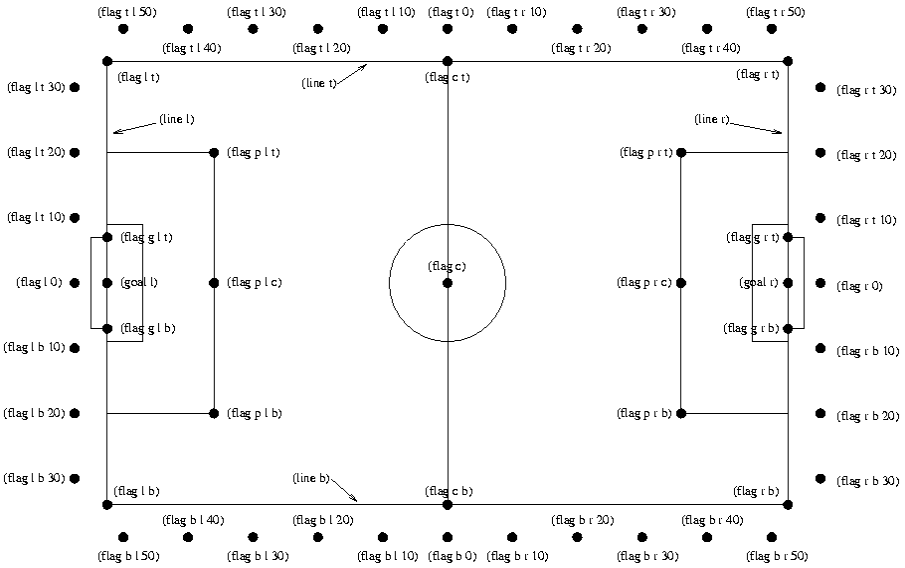

Figura 56: Localização e nomes dos marcos de orientação no Soccer Server.

As bandeiras *(goal r)* e *(goal l)* correspondem a bandeiras virtuais localizadas no centro das balizas direita e esquerda. *(flag g l b), (flag g l t), (flag g r b)* e *(flag g r t)* correspondem aos postes virtuais das duas balizas. *(flag c)* corresponde a uma bandeira virtual no centro do campo. Existem ainda bandeiras colocadas nos limites do campo e das áreas. As restantes bandeiras estão colocadas fora do campo e a 5 metros do limite deste, espaçadas entre elas de 10 metros. Por exemplo, *(flag b l 30)* está 5 metros abaixo da linha lateral inferior do campo, 30 metros à esquerda do centro do campo. Existem ainda 4 linhas visíveis que limitam o campo. Estas correspondem a *(line t), (line b), (line r)* e *(line l)* que limitam o campo respectivamente em cima, em baixo, à direita e à esquerda. Cada bandeira é identificada por uma etiqueta que é usada para a sua identificação durante a comunicação simulador-jogador.

Cada jogo de futebol robótico simulado tem uma duração de 10 minutos divididos em duas partes de 5 minutos, correspondentes a 3000 ciclos de simulação cada[51]. Em caso de empate no final do tempo regulamentar, seguem-se prolongamentos sucessivos de 5 minutos (3000 ciclos) seguindo a regra do golo dourado[52]. Em competições oficiais, dependendo do tipo de jogo disputado, o jogo pode ser terminado no final do tempo regulamentar, com um empate, mesmo sem recurso a prolongamento.

8.3.2.2 Os Árbitros Artificial e Humano

Durante cada partida existe um determinado conjunto de regras que é garantido pelo árbitro automático contido no servidor que comunica com os jogadores através de mensagens, e por um árbitro humano designado para supervisionar o jogo.

O árbitro artificial consegue arbitrar o jogo totalmente sem necessidade de auxílio do árbitro humano e em geral, em jogos entre equipas de boa qualidade e com *fair-play*, o árbitro humano é dispensável. No entanto, faltas tais como obstruções ou anti-jogo são muito difíceis de ser julgadas pelo árbitro artificial pois dizem respeito a intenções dos jogadores. Para resolver este tipo de situações, o *soccer monitor* providencia uma interface que permite a um árbitro humano efectuar duas operações parametrizáveis de supervisão do jogo: marcar uma falta a favor de uma das equipas num dado ponto do campo e "lançar a bola ao solo" num dado ponto do campo. No anexo 4 são apresentadas as principais características do árbitro artificial, as regras que este é responsável por garantir, as mensagens que este agente pode enviar aos jogadores e a sua respectiva sintaxe.

---

[51] Isto considerando os parâmetros por defeito actuais do simulador, nomeadamente ciclos de simulação de 100 mili-segundos.

[52] Também designada vulgarmente por morte-súbita, isto é, a equipa que marcar o primeiro golo no prolongamento vence de imediato o jogo.

Todos os anos, as capacidades do árbitro artificial são estendidas de forma a torná-lo cada vez mais autónomo e evitar a necessidade de existir intervenção humana durante o jogo. Neste sentido, em 2001, o árbitro passou a realizar a marcação de bolas ao solo no caso de uma equipa não recolocar rapidamente a bola em jogo ou de o guarda-redes de uma equipa efectuar vários *moves* em sequência antes de repor a bola. Em 2002, o árbitro foi estendido de forma a marcar livres no caso de atrasos de bola para o guarda-redes ou marcação de livres em que um jogador passa a bola para si mesmo e prossegue com ela controlada.

Ao longo dos próximos anos é previsível que o árbitro artificial seja ainda mais desenvolvido de forma à intervenção humana na arbitragem dos jogos ser totalmente desnecessária. É previsível também a separação deste módulo do servidor num agente autónomo com capacidades de análise e controlo do jogo. Isto permitirá aumentar a modularidade do sistema e permitirá que diferentes grupos de investigação desenvolvam os seus próprios árbitros. Desta forma, os diferentes jogos oficiais poderão ser arbitrados por árbitros artificiais construídos por diferentes universidades e países, à semelhança do que se passa no futebol real.

8.3.3 Protocolos de Comunicação com o Servidor

De forma a poderem participar no jogo simulado, os clientes têm de possuir capacidades que lhes permitam seguir um determinado conjunto de protocolos de comunicação com o servidor. De entre estes protocolos destacam-se os protocolos de conexão, acção e percepção.

8.3.3.1 Protocolo de Conexão dos Clientes

O protocolo de conexão (tabela 10) permite a um cliente conectar-se, desconectar-se ou reconectar-se ao simulador.

Conexão (Cliente para o Servidor)	***Conexão (Resposta do Servidor)***
*(***init** *<NomeEquipa> [(version <NumVer>)] [(goalie)])* *<NomeEquipa> ::= (-|_|a-z|A-Z|0-9)* *<NumVer> ::= versão do protocolo de comunicação a utilizar (p.e. 7.0)*	*(***init** *<Lado> <Unum> <ModoJogo>)* *<Lado> ::= l|r* *<Unum> ::= 1-11* *<ModoJogo> ::= um dos modos de jogo válido* *(***error** *no_more_team_or_player_or_goalie)*
Reconexão (Cliente para o Servidor)	***Reconexão (Resposta do Servidor)***
*(***reconnect** *<NomeEquipa> <Unum>)* *<NomeEquipa> ::= (-|_|a-z|A-Z|0-9)*	*(***init** *<Lado> <Unum> <ModoJogo>)* *<Lado> ::= l|r* *<Unum> ::= 1-11* *<ModoJogo> ::= um dos modos de jogo válido* *(***error** *no_more_team_or_player)* *(***error** *reconnect)*
Disconexão (Cliente para o Servidor)	***Disconexão (Resposta do Servidor)***
*(***bye***)*	

Tabela 10: Protocolo de Conexão dos Clientes ao Simulador

Como o *soccerserver* é um sistema em constante evolução, permite a conexão de clientes que utilizam versões do protocolo distintas, permitindo desta forma que clientes desenvolvidos para versões anteriores do *soccerserver* se possam conectar ao sistema. No caso do cliente se conectar com um protocolo superior ou igual ao 7.0, então receberá adicionalmente a informação relativa aos parâmetros do servidor[53] e aos parâmetros dos jogadores heterogéneos disponíveis[54].

8.3.3.2 Protocolo de Percepção dos Clientes

A percepção dos clientes está dividida em três partes distintas: percepção auditiva (***hear***), visual (***see***) e sensorial (***sense_body***). A tabela 11 descreve em detalhe o protocolo utilizado pelo servidor para enviar estes três tipos de percepção aos clientes.

Percepção (Servidor para o Cliente)
*(***hear** *<Tempo> <Emissor> <Mensagem>)* *<Tempo> ::= ciclo de simulação do simulador* *<Emissor> ::= online_coach_left | online_coach_right | referee | self | <Direcção>* *<Direcção> ::= -180..180* *<Mensagem> ::= [string]*
*(***see** *<Tempo> <InfoObj>)* *<Tempo> ::= ciclo de simulação do simulador* *<InfoObj> ::= (<NomeObj> <Distância> <Direcção> <DistVar> <DirVar> <DirCorpo> <DirCabeça>) | (<NomeObj> <Distância> <Direcção> <DistVar> <DirVar>) | (<NomeObj> <Distância> <Direcção>) | (<NomeObj> <Direcção>)* *<NomeObj> ::= (p <NomeEquipa> [<Unum> [goalie]]]) | (b) | g [l|r] | (l [l|r|t|b]) | (f c) | (f [l|c|r] [t|b]) | (f p [l|r] [t|c|b]) | (f g [l|r] [t|b])| (f [l|r|t|b] 0) | (f [t|b] [l|r] [10|20|30|40|50]) | (f [l|r] [t|b] [10|20|30])* *<Distância> ::= Real positivo* *<Direcção> ::=[-180.0 .. 180.0]* *<DistVar> ::= Real* *<DirVar> ::= Real* *<DirCorpo> ::= [-180.0, 180.0]* *<DirCabeça>::= [-180.0, 180.0]* *<NomeEquipa> ::= [string]* *<Unum> ::= [1..11]*
*(***sense_body** *<Tempo> (*view_mode *{high|low} {narrow|normal|wide}) (*stamina *<Energia> <Esforço>) (*speed *<ValorVel> <DirVel>*) (head_angle *<DirCabeça>*) (kick *<ContagemKicks>*) (dash *<ContagemDashes>*) (turn *<ContagemTurns>*) (say *<ContagemSays>*) (turn_neck *<ContagemTurnNecks>*) (catch *<ContagemCatchs>*) (move *<ContagemMoves>*) (change_view *<ContagemChangeViews>*))* *<Tempo> ::= ciclo de simulação do simulador*

[53] Antes da versão 7.0 os clientes tinham de possuir um ficheiro de configuração com os parâmetros utilizados pelo servidor.

[54] Antes da versão 7.0 os jogadores eram homogéneos.

<Energia> ::=[0..4000]
<Esforço> ::=[0..1.0]
<ValorVel> ::= real positivo
<DirVel> ::=[-180.0 .. 180.0]
<DirCabeça> ::=[-180.0 .. 180.0]
<Contagem> ::= inteiro positivo (para todas as contagens)*

Tabela 11: Protocolo de Percepção dos Clientes

O protocolo de percepção dos clientes é relativamente simples, pois não é previsto que os clientes enviem qualquer tipo de resposta ao servidor após a recepção de informação sensorial.

8.3.3.3 Protocolo de Acção dos Clientes

O protocolo de acção dos clientes especifica a sintaxe das mensagens de comando que os clientes estão autorizados a enviar para o servidor e as possíveis respostas do servidor a estas mensagens. Este protocolo é em geral muito simples, pois a maioria das mensagens enviadas pelos clientes não implicam a ocorrência de resposta por parte do servidor. O protocolo encontra-se representado na tabela 12.

Acção (Cliente para o Servidor)
*(***dash** *<Potência>)* *<Potência> ::= [-100, 100]*
*(***turn** *<Momento>)* *<Momento> ::= [-180, 180]*
*(***move** *<PosX> <PosY>)* *<PosX> ::= [-52.5, 52.5]* *<PosY> ::= [-34.0, 34.0]*
*(***kick** *<Potência> <Direcção>)* *<Potência> ::= [-100, 100]* *<Direcção> ::= [-180, 180]*
*(***tackle** *<Potência>)* *<Potência> ::= [0, 100]*
*(***catch** *<Direcção>)* *<Direcção> ::= [-180, 180]*
*(***turn_neck** *<Direcção>)* *<Direcção> ::= [-180, 180]*
*(***change_view** *<Abertura> <Qualidade>)* *<Abertura> ::= narrow | normal | wide* *<Qualidade> ::= high | low*
*(***attentionto** *<Equipa> <Unum>)* | *(attentionto off)* *<Equipa> ::= opp | our | l | r | left | right | <Nome_Equipa>*
*(***say** *<Mensagem>)* *<Mensagem> ::= "Texto"*
*(***pointto** *<Distância> <Direcção>)* | *(pointo off)* *<Distância> ::= Inteiro* *<Direcção> ::= [-180, 180]*

Acção (Resposta do Servidor)
*(***error** *unknown_command)* *(***error** *ilegal_command_form)* *(***score** <*Tempo*> <*GolosEquipa*> <*GolosOponente*>*) para comandos (***score***)* *(***sense_body** <*Tempo*> *(*view_mode *{high|low} {narrow|normal|wide}) (*stamina <*Energia*> <*Esforço*>*)* *(*speed <*ValorVel*> <*DirVel*>) (head_angle <*DirCabeça*>) (kick <*ContagemKicks*>) (dash <*ContagemDashes*>) (turn <*ContagemTurns*>) (say <*ContagemSays*>) (turn_neck <*ContagemTurnNecks*>) (catch <*ContagemCatchs*>) (move <*ContagemMoves*>) (change_view <*ContagemChangeViews*>)) *para comandos (***sense_body***)*

Tabela 12: Protocolo de Acção dos Clientes

Os clientes enviam comandos que o simulador executa sem enviar qualquer resposta aos clientes. No caso do comando ser desconhecido ou os seus parâmetros serem ilegais, o simulador responde com uma mensagem de erro. Só existem dois casos em que o simulador responde ao comando enviado pelos clientes que corresponde ao comando (***score***) a que o simulador responde com o tempo e resultado do jogo e o comando (***sense_body***) a que o simulador responde com informação sensorial física para o agente.

8.3.4 Percepção dos Agentes

No futebol robótico simulado, como foi referido anteriormente, os agentes têm três tipos de sensores. O sensor auditivo detecta mensagens enviadas pelo árbitro, treinador, colegas de equipa e adversários. O sensor visual detecta informação visual disponível que inclui distâncias e direcções dos objectos e jogadores que se encontram no campo de visão do agente. O sensor físico detecta o estado do agente, incluindo a sua energia, velocidade e ângulo do pescoço relativamente ao corpo. Utilizando correctamente estes três sensores, o agente deve ser capaz de construir uma imagem clara das ocorrências no campo de jogo.

8.3.4.1 Informação Visual

A informação visual desempenha um papel primordial no futebol robótico simulado. O sensor visual dos agentes permite-lhes obter informação sobre a direcção e distância dos objectos que se encontram no seu campo visual em cada instante. O sensor visual desempenha também o papel de sensor de proximidade, dando aos agentes informação sobre os objectos que se encontram muito próximos mas atrás do agente. A informação visual é enviada automaticamente pelo servidor periodicamente cada *send_step* ou *send_step/2* dependendo do modo de visualização seleccionado pelo jogador.

É importante perceber que a informação enviada a cada agente é relativa à perspectiva desse agente, e como resultado disto um dado agente não se pode aperceber directamente da sua posição global ou da posição global dos outros jogadores ou da bola. Para tal, a informação relativa recebida deve ser convertida em informação absoluta. Como auxílio à realização desta conversão, conforme demonstrado na figura anterior, diversos marcos e

linhas encontram-se colocados no campo. Combinando as posições globais conhecidas destes marcos com as suas posições relativas que se encontram incluídas nas mensagens de visualização do agente, é possível ao agente aferir a sua posição global no campo e consequentemente determinar as posições relativas dos outros agentes e da bola.

Um jogador pode controlar a frequência, alcance e qualidade da informação visual que lhe é enviada pelo simulador. A frequência por defeito de envio da informação visual é controlada pelo parâmetro *send_step* que corresponde a 150 ms. No entanto, o agente pode trocar frequência por qualidade da informação visual e abertura do seu cone de visão. Para o fazer, pode ajustar a sua qualidade de visualização e a sua abertura de visualização, *ViewQuality* e *ViewWidth* respectivamente. A tabela 13 demonstra a frequência visual dos agentes tendo em conta o modo visual seleccionado. Esta frequência visual é calculada da seguinte forma:

frequência_visual = *send_step* * ***factor_qualidade*** * ***factor_abertura***

em que **factor_qualidade** é 0.5 se *ViewWidth* for ***narrow***, 1 se for ***normal*** e 2 se for ***wide*** e **factor_abertura** é 1 se *ViewQuality* for ***high*** e 0.5 se for ***low***. O cone de visão do agente é determinado pelo parâmetro do servidor *visible_angle* que representa a abertura em graus do cone de visão normal do agente (que corresponde a 90º actualmente). O ângulo de visualização é calculado da seguinte forma:

ângulo_visualização = *visible_angle* * ***factor_abertura***

Um agente pode ainda ver (sentir) objectos que estão até *visible_distance* do seu corpo embora só perceba o seu tipo (bola, jogador, baliza, bandeira, etc.) sem receber informação acerca da sua identificação (qual o jogador, qual a bandeira, etc.).

Width / ***Quality***	***Narrow***	***Normal***	***Wide***
High	*send_step/2* *75 ms*	*send_step* *150 ms*	*send_step*2* *300 ms*
Low	*send_step/4* *37.5 ms*	*send_step/2* *75 ms*	*send_step* *150 ms*

Tabela 13: Frequência visual dos agentes de acordo com o seu modo visual

A informação visual é enviada pelo servidor aos jogadores no formato seguinte:

(**see** <*Tempo*> <*InfoObj*>)

onde:

<*Tempo*> *::= Inteiro (e corresponde ao ciclo do simulador)*

<*InfoObj*> *::= (*<*NomeObj*> <*Distância*> <*Direcção*> *[*<*DistVar*> <*DirVar*>

[<*DirCorpo*> <*DirCabeça*>*]]) | (*<*NomeObj*> <*Direcção*>*)*

dependendo do tipo de visão seleccionado e onde:

<*NomeObj*> *::= (p* <*NomeEquipa*> *[*<*UNum*> *[goalie]]) |*

(b) | g [l|r] | (l [l|r|t|b]) |

(f c) | (f [l|c|r] [t|b]) | (f p [l|r] [t|c|b]) | (f g [l|r] [t|b])|

(f [l|r|t|b] 0) | (f [t|b] [l|r] [10|20|30|40|50]) | (f [l|r] [t|b] [10|20|30])

<Distância> ::= Real positivo

<Direcção> ::=[-180.0 .. 180.0]

<DistVar> ::= Real

<DirVar> ::= Real

<DirCorpo> ::= [-180.0 .. 180.0]

<DirCabeça>::= [-180.0 .. 180.0]

<NomeEquipa> ::= [string]

<Unum> ::= [1 .. 11]

Esta informação visual é fornecida para todos os objectos visíveis, isto é, aqueles que estão no cone de visão do jogador. A quantidade de informação fornecida para cada objecto, depende do tipo e distância do objecto e da qualidade de visão seleccionada pelo jogador. Quando a qualidade de visão for ***low***, as únicas informações fornecidas são o nome e direcção do objecto. Se a qualidade for ***high***, a distância ao objecto é também fornecida e dependendo do tipo de objecto outra informação pode ainda ser fornecida como seja a direcção do corpo ou do pescoço no caso de um jogador.

A informação relativa a *<Distância>*, *<Direcção>*, *<DistVar>* e *<DirVar>* é calculada da seguinte forma [Soccerserver, 2001]:

$p_{rx} = p_{xt} - p_{xo}$

$p_{ry} = p_{yt} - p_{yo}$

$v_{rx} = v_{xt} - v_{xo}$

$v_{ry} = v_{yt} - v_{yo}$

$<Distância> = \sqrt{p_{rx}^2 + p_{ry}^2}$

$<Direcção> = arctan\ (p_{ry}/p_{rx}) - Dir_o$

$e_{rx} = p_{rx} / <Distância>$

$e_{ry} = p_{ry} / <Distância>$

$<DistVar> = (v_{rx} * e_{rx}) + (v_{ry} * e_{ry})$

$<DirVar> = ((-(v_{rx} * e_{rx}) + (v_{ry} * e_{ry}))/<Distância>)*(180/\pi)$

Onde (p_{xt}, p_{yt}) é a posição absoluta do objecto visualizado, (p_{xo}, p_{yo}) é a posição absoluta do agente que está a visualizar o objecto, (v_{xt}, v_{yt}) é a velocidade absoluta do objecto visualizado, (v_{xo}, v_{yo}) é a velocidade absoluta do agente que está a visualizar o objecto e Dir_o é a direcção absoluta para a qual o agente se encontra direccionado. Desta forma, (p_{rx}, p_{ry}) e (v_{rx}, v_{ry}) são respectivamente a posição relativa e a velocidade relativa do objecto visualizado e (e_{rx}, e_{ry}) é um vector unitário que é paralelo ao vector (p_{rx}, p_{ry}).

Os parâmetros *<DirCorpo> e <DirCabeça>* só são incluídos no caso do objecto observado ser um jogador e correspondem à direcção relativa do corpo e cabeça do jogador, observado relativamente ao observador.

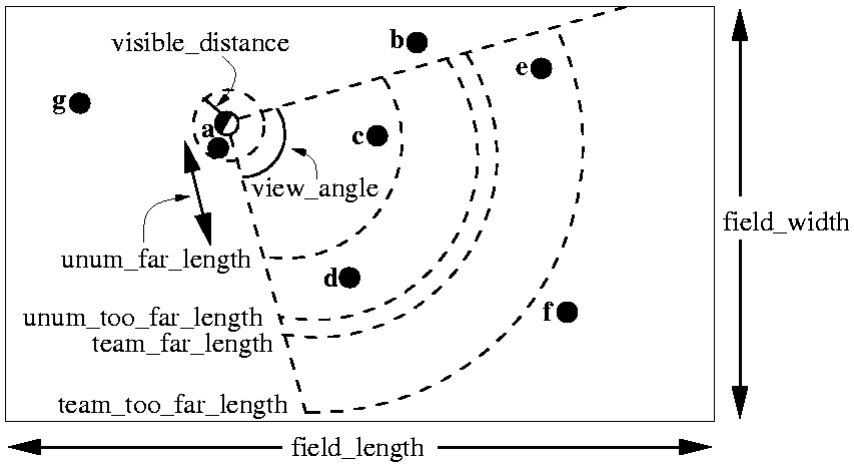

Figura 57: Percepção Visual dos agentes [Stone, 1998].

A figura 57 demonstra a percepção visual dos agentes. O Agente cuja percepção visual está a ser ilustrada é representado através de dois semicírculos (em que o semicírculo claro corresponde à frente do agente). Os círculos negros correspondem a outros jogadores no campo. Só os objectos dentro do ângulo de visão (*view_angle*) ou a uma distância inferior a *visible_distance* são visíveis. Os parâmetros *unum_far_length, unum_too_far_length, team_far_length* e *team_too_far_length* afectam a precisão e tipo de informação visual enviada ao agente.

A informação visual enviada, pelo servidor pode ser sumariada da seguinte forma:

- **Objectos Estáticos (bandeiras, linhas e balizas):** nome do objecto, distância relativa e direcção. No caso das linhas, a distância corresponde à distância ao ponto da linha onde o bissector do cone de visão do jogador cruza essa linha e a direcção é o ângulo entre a linha e esse bissector.
- **Jogadores:** A informação depende da distância do observador ao jogador da forma seguinte:

  Se *<Distância>* ≤ *unum_far_length*, então quer o número quer a equipa do jogador são incluídas. Os valores de *<Distância>, <Direcção>, <DistVar>, <DirVar>, <DirCorpo>* e *<DirCabeça>* são todos incluídos.

  Se *unum_far_length* < *<Distância>* ≤ *unum_too_far_length* = *team_far_length*, então a equipa do jogador e os valores de *<Distância>, <Direcção>*, serão visíveis. No entanto, a probabilidade de que o número do jogador seja visualizado decresce linearmente de 1 par 0 entre *unum_far_length* e *unum_too_far_length.* O mesmo acontece com a probabilidade de serem incluídos os valores de *<DistVar>, <DirVar>, <DirCorpo>* e *<DirCabeça>*.

Se *unum_too_far_length* = *team_far_length* < <*Distância*> ≤ *team_too_far_length*, então a informação relativa ao número do jogador e os valores de <*DistVar*>, <*DirVar*>, <*DirCorpo*> e <*DirCabeça*> nunca serão dados. A probabilidade de a equipa do jogador ser fornecida na informação visual, decresce linearmente de 1 a 0 entre *team_far_length* e *team_too_far_length.*

Se <*Distância*> > *team_too_far_length* então o jogador será identificado como um anónimo sem qualquer informação do seu número ou equipa.

- **Bola:** Para a bola a situação é similar aos jogadores, isto é:

  Se <*Distância*> ≤ *unum_far_length*, então os valores de <*Distância*>, <*Direcção*>, <*DistVar*>, <*DirVar*> são todos incluídos.

  Se *unum_far_length* < <*Distância*> ≤ *unum_too_far_length* = *team_far_length*, então os valores de <*Distância*>, <*Direcção*> serão visíveis e a probabilidade de que <*DistVar*> e <*DirVar*> sejam visíveis decresce linearmente de 1 para 0 entre *unum_far_length* e *unum_too_far_length.*

  Se <*Distância*> > *unum_too_far_length* = *team_far_length*, então a informação relativa a <*DistVar*> e <*DirVar*> nunca será dada.

8.3.4.2 Informação Auditiva

A informação auditiva é de extrema importância, pois todas as mensagens enviadas pelo árbitro são deste tipo. Desta forma, um jogador não pode desempenhar correctamente o seu papel na equipa sem ser capaz de interpretar correctamente as mensagens auditivas.

Para além das mensagens do árbitro, os jogadores recebem também por via auditiva, mensagens do seu treinador (caso este exista) e dos outros jogadores (colegas de equipa e adversários). O formato para as mensagens auditivas enviadas pelo simulador é:

(**hear** <*Tempo*> <*Direcção*> <*Equipa*> *[*<*Emissor*>*]* <*Mensagem*>) ou

(**hear** <*Tempo*> <*Equipa*> *[*<*Emissor*>*]*)

onde:

<*Tempo*> *::= Inteiro*

<*Direcção*> *::= Real*

<*NomeEquipa*> *::= string*

<*Emissor*> *::= self* | *referee* | *online_coach_left* | *online_coach_right* | <*Direcção*>

<*Mensagem*> *::= string*

em que <*Tempo*> indica o tempo corrente. <*Emissor*> pode ser *self* – próprio jogador, *referee* – árbitro, *online_coach_left* ou *online_coach_right* – um dos treinadores ou <*Direcção*> – direcção relativa do jogador que enviou a mensagem.

Diversos parâmetros permitem configurar esquemas de comunicação diferentes no simulador. Os parâmetros principais permitem configurar a distância máxima de

comunicação (*audio_cut_dist*), o número máximo de mensagens que é possível ouvir num dado número de ciclos (*hear_max*, *hear_inc* e *hear_decay*) e a dimensão máxima das mensagens (*say_msg_size*).

Para além da limitação relativa à capacidade auditiva, o servidor implementa também uma restrição relativa ao alcance espacial das mensagens. Uma mensagem só pode ser escutada por jogadores que estejam a uma distância máxima de *audio_cut_dist* do emissor. A distância máxima utilizada em competições oficiais tem sido de 50 metros tornando desta forma, por exemplo, muito difícil a um guarda-redes, escutar qualquer mensagem enviada por jogadores que estejam na metade oposta do campo. As mensagens enviadas pelos treinadores e árbitro não têm limitações de alcance.

Diversas alterações têm sido introduzidas no modelo de comunicação ao longo dos anos. A tabela 14 sumaria os parâmetros de configuração do modelo de comunicações e as alterações realizadas nestes parâmetros de 2001 para 2002.

Parâmetro / ***Competição***	***Dimensão Máxima***	***Alcance Máximo***	***Instante de Recepção***	***Frequência Auditiva***	***Identificação do Emissor***
Antes de 2001	*512 Bytes*	*50 metros*	*No mesmo ciclo*	*1 mensagem cada 2 ciclos*	*Equipa e direcção do emissor*
A partir de 2002	*10 Bytes*	*50 metros*	*Ciclo seguinte à emissão*	*1 mensagem por ciclo*	*Equipa e número do emissor*

Tabela 14: Alterações no Modelo de Comunicações de 2001 para 2002

Na tabela é visível que a dimensão máxima das mensagens sofreu um decréscimo considerável de 2001 para 2002 (512 bytes para unicamente 10 bytes) e as mensagens passaram unicamente a ser ouvidas pelos agentes no ciclo seguinte à sua emissão. Como compensação para este decréscimo foi duplicada a frequência auditiva das mensagens e foi introduzida a identificação do emissor (simulando o reconhecimento de vozes humano), ou seja:

<Emissor> ::= self | referee | online_coach_left | online_coach_right | <Number>

Estes novos parâmetros de comunicação tornaram muito mais complicada a decisão de quando e o quê comunicar. Exigiram também a implementação de algoritmos de compressão da informação a comunicar por parte das equipas.

8.3.4.3 Informação Física

Para além da percepção visual e auditiva, o simulador envia também aos agentes percepção física, correspondente ao estado do agente. Esta informação é automaticamente enviada aos jogadores a cada *sense_body_step*, actualmente definido como 100 mili-segundos. O formato da mensagem de envio de informação física é:

(**sense_body** *Tempo*
(view_mode *Qualidade Abertura*) (stamina *Energia Esforço*) (speed *ValorVel DirVel*)

(head_angle *DirCabeça*) (kick *ContagemKicks*) (dash *ContagemDashes*)
(turn *ContagemTurns*) (say *ContagemSays*) (turn_neck *ContagemTurnNecks*)
(catch *ContagemCatchs*) (move *ContagemMoves*) (change_view *ContagemCViews*))

A *Qualidade* e *Abertura* indicam respectivamente a qualidade (*high* ou *low*) e abertura (*narrow*, *normal* ou *wide*) actuais da visão do agente.

Os parâmetros *Energia* e *Esforço* referem-se à energia (*Stamina*) e esforço actuais do agente. Indicações da velocidade absoluta e respectiva direcção e do ângulo relativo entre a cabeça e o corpo do agente são fornecidos nos parâmetros seguintes (*ValorVel DirVel* e *DirCabeça*).

Os 8 parâmetros finais dizem respeito à contagem de comandos enviados pelo agente que foram efectivamente executados pelo simulador. Por exemplo, *ContagemKicks* = 50, indicará que o agente executou até ao momento, com sucesso, 50 comandos do tipo *kick*. Esta indicação é útil para detectar problemas de rede, problemas de sincronização com o servidor ou de falta de capacidade de processamento.

8.3.5 Acção dos Agentes

Os agentes da liga de simulação do RoboCup têm à sua disposição um vasto leque de comandos parametrizáveis, tornando o número de acções possível a cada agente em cada instante neste domínio, virtualmente infinito. Estas acções básicas encontram-se sumariadas na tabela 15.

Comando	***Parâmetros***	***Limitações***
(**dash** <*Potência*>)	<*Potência*> *::= [-100, 100]*	*Clientes podem enviar unicamente um comando do tipo dash, turn, move, kick, tackle ou catch em cada ciclo* *Potências negativas consumem o dobro da energia*
(**turn** <*Momento*>)	<*Momento*> *::= [-180, 180]*	*Clientes podem enviar unicamente um comando do tipo dash, turn, move, kick, tackle ou catch em cada ciclo*
(**move** <*PosX*> <*PosY*>)	<*PosX*> *::= [-52.5, 52.5]* <*PosY*> *::= [-34.0, 34.0]*	*Este comando só pode ser executado no início de cada metade do jogo ou após um golo de uma equipa ou pelo guarda-redes após agarrar a bola.*
(**kick** <*Potência*> <*Direcção*>)	<*Potência*> *::= [-100, 100]* <*Direcção*> *::= [-180, 180]*	*A bola deve estar suficientemente próxima do jogador para que este a possa chutar*
(**tackle** <*Potência*>)	<*Potência*> *::= [0, 100]*	*O comando tackle só pode ser executado em servidores a partir da versão 8.x* *Após executar um comando tackle, o jogador fica imóvel durante alguns ciclos*
(**catch** <*Direcção*>)	<*Direcção*> *::= [-180, 180]*	*O comando catch só pode ser executado pelo guarda-redes.* *Para o guarda-redes conseguir agarrar a bola, esta tem de estar suficientemente próxima.*

(**turn_neck** <*Direcção*>)	<*Direcção*> *::= [-180, 180]*	
(**change_view** <*Abertura*> <*Qualidade*>)	<*Abertura*> *::= narrow | normal | wide* <*Qualidade*> *::= high | low*	
(**attentionto** <*Equipa*> <*Unum*>) | (**attentionto** *off*)	<*Equipa*> *::= opp | our | l | r | left | right | <Nome_Equipa>*	
(**say** <*Mensagem*>)	<*Mensagem*> *::= "Texto"*	*Texto com uma dimensão máxima de 10 bytes. Antes da versão 8.x, a dimensão máxima do texto era de 512 bytes.*
(**pointto** <*Distância*> <*Direcção*>) | (**pointo** *off*)	<*Distância*> *::= Inteiro* <*Direcção*> *::= [-180, 180]*	

Tabela 15: Acções Possíveis para os Agentes na Liga de Simulação do RoboCup

As acções podem-se decompor em quatro tipos principais: movimento (***dash***, ***turn*** e ***move***); interacção com a bola (***kick***, ***tackle*** e ***catch***); controle da percepção (***turn_neck***, ***attentionto***, ***change_view***) e comunicação (***say***, ***pointto***). Uma restrição importante consiste no facto de os agentes poderem enviar unicamente um comando do tipo movimento ou interacção com bola por ciclo. Desta forma é impossível, por exemplo, acelerar e chutar a bola ao mesmo tempo. No caso de um agente enviar mais do que um comando primário (movimento ou interacção com a bola) para o servidor, este selecciona um comando aleatoriamente para ser executado.

Embora as acções básicas disponíveis aos agentes na liga de simulação do RoboCup tenham permanecido relativamente estáveis, ao longo dos anos algumas modificações têm sido introduzidas. De entre estas destacam-se a introdução do pescoço flexível e consequente comando ***turn_neck*** em 1999; a introdução da acção ***tackle*** em 2002; a introdução do comando de controle de percepção ***attentionto*** em 2002 e a inclusão do comando ***pointto*** também em 2002.

Uma outra alteração importante relativamente à capacidade de acção dos agentes foi introduzida em 2001 com a inclusão de jogadores heterogéneos. Estes jogadores possuem actuadores com características diferentes e consequentemente capacidades de acção distintas.

8.3.5.1 Movimento – Move

O simulador permite aos agentes enviar três comandos com o objectivo de movimentar os jogadores que controlam. Estes comandos são:

(**dash** <*Potência*>)

(**turn** <*Momento*>)

(**move** <*PosX*> <*PosY*>)

em que <*Potência*> $\subset$ *[minpower, maxpower]*, <*Momento*> $\subset$ *[-180, 180]*, <*PosX*> $\subset$ *[-52.5, 52.5]* e <*PosY*> $\subset$ *[-34.0, 34.0]*.

O Comando ***move*** pode ser utilizado para posicionar directamente a equipa no campo. Permite a um jogador posicionar-se exactamente num dado ponto *(<PosX>, <PosY>)* do campo. No entanto não pode ser utilizado, em condições normais, durante o jogo. Só pode ser utilizado durante os modos *before_kick_off, goal_l* e *goal_r,* ou seja antes do início de uma das partes do jogo e após um golo de uma das equipas. Durante estes modos, os jogadores podem ser posicionados nas suas metades do campo (*x<0*) e podem ser movimentados utilizando este comando, o número de vezes desejado.

O segundo objectivo do comando ***move*** consiste em permitir ao guarda-redes movimentar-se na sua grande área, após ter agarrado a bola (utilizando um comando ***catch***). O guarda-redes pode movimentar-se em conjunto com a bola *goalie_max_moves* (cujo valor por defeito é 2), antes de chutar a bola.

8.3.5.2 Movimento – Aceleração e Energia

O comando ***dash*** é utilizado para acelerar um jogador na direcção do seu corpo com uma dada potência. *Potência* é utilizado como parâmetro, sendo os seus valores mínimo e máximo parâmetros do servidor, actualmente configurados para –100 e 100 respectivamente. O simulador evita que os jogadores possam movimentar-se sempre à máxima velocidade (*player_speed_max*), atribuindo-lhes uma energia limitada. Esta energia (*stamina*) é consumida sempre que o jogador utiliza um comando ***dash***[55]. No início (e no intervalo) de cada jogo, a energia é colocada ao máximo (*stamina_max*, actualmente 4000). Sempre que um jogador acelera no sentido positivo (*Potência > 0*), a sua energia é diminuída de *Potência*. Acelerar no sentido negativo (*Potência < 0*), é mais dispendioso em termos de energia, consumindo *2*Potência*.

A energia de cada jogador é modelada utilizando três parâmetros:

- ***Energia***: representa o valor da energia (*stamina*) corrente do jogador e deve ter um valor compreendido entre *[0, stamina_max]*. Um comando ***dash*** não pode ter uma *Potência* superior ao valor de *Energia*;
- ***Esforço***: representa a eficiência dos movimentos do jogador e tem um valor compreendido entre *[effort_min, effort_max]*. Uma baixa capacidade de esforço significa que a *Potência_Real* dos comandos executados pelo jogador vai ser reduzida, ou seja: *Potência_Real = Esforço * Potência*;
- ***Recuperação***: Define a taxa de recuperação de energia dos jogadores. Está contida no intervalo *[recover_min, 1.0]*.

---

[55] Embora actualmente, o comando dash seja o único a consumir energia, tem sido sugerido por diversos investigadores a extensão deste princípio a outros comandos tais como o kick e o turn.

A energia de um jogador é decrementada sempre que este executa um comando ***dash*** mas é restaurada ligeiramente no final de cada ciclo de simulação. Todos os ciclos, se a *Energia* do jogador for inferior a um determinado limiar, o simulador decrementa o *Esforço* e a capacidade de *Recuperação* do jogador. Se o valor da *Energia* for superior a um outro limiar, o *Esforço* é incrementado. A *Recuperação* nunca é incrementada durante o decurso normal do jogo[56]. O algoritmo 1 demonstra em detalhe o modelo completo de cálculo da energia aplicado pelo simulador em cada ciclo.

```
// Redução da energia de acordo com a Potência de dash
SE (comando dash recebido) ENTÃO
       SE Potência > 0 ENTÃO Energia = Energia – Potência
                       SENÃO Energia = Energia – 2*|Potência|
// Redução da capacidade de recuperação quando a energia está abaixo do limiar
de decremento da recuperação
SE Energia <= recover_dec_thr * stamina_max ENTÃO
       SE Recuperação > recover_min ENTÃO Recuperação = Recuperação – recover_dec
       Recuperação = Max(recover_min, Recuperação)
// Redução da capacidade de esforço quando a energia está abaixo do limiar de
decremento do esforço
SE Energia <= effort_dec_thr * stamina_max ENTÃO
       SE Esforço > effort_min ENTÃO Esforço = Esforço – effort_dec
       Esforço = Max(effort_min, Esforço)
// Incremento da capacidade de esforço quando a energia está acima do limiar de
incremento do esforço
SE Energia >= effort_inc_thr * stamina_max ENTÃO
       SE Esforço < effort_max ENTÃO Esforço = Esforço + effort_inc
       Esforço = Min(effort_max, Esforço)
// Restauro de energia de acordo com o valor da Recuperação
Energia = Min (stamina_max, Energia + Recuperação * stamina_inc_max)
```

Algoritmo 1: Algoritmo do Modelo de Energia do Simulador

Quando um agente executa um comando ***dash***, o simulador utiliza a Potência Real do comando de forma a calcular a aceleração a aplicar ao jogador na transição do ciclo actual de simulação para o ciclo seguinte:

$$\left(a_x^t, a_y^t\right) = Potência\_Real * dash\_power\_rate * (\cos(\theta^t), \sin(\theta^t))$$

onde *dash_power_rate* é um parâmetro do servidor utilizado para determinar a dimensão do vector aceleração aplicado ao jogador e θ^t é a direcção do corpo do jogador no ciclo de simulação *t*. O vector de aceleração é então normalizado para um comprimento absoluto máximo de *player_accel_max*, após o que é adicionado vectorialmente à velocidade corrente do jogador. A velocidade resultante é então normalizada para um máximo de

[56] Embora seja restaurada a 1.0 no início de cada parte do jogo.

player_speed_max e é adicionada vectorialmente à posição do jogador para calcular a sua nova posição. Desta forma, *player_speed_max*, corresponde aproximadamente à máxima distância que um jogador pode percorrer num ciclo de simulação[57]. O algoritmo 2 demonstra com maior detalhe o modelo de movimento do simulador:

```
// Normalização da aceleração
        |a_t| = Max(|a_t|, player_accel_max)
// Cálculo da velocidade
        v_t = v_t + a_t
// Normalização da velocidade
        |v_t| = Max(|v_t|, player_speed_max)
// Adição de Ruído à velocidade
        v_t = v_t + Ruído
// Cálculo da nova posição do Jogador
        p_t+1 = p_t + v_t
// Redução da Velocidade do jogoador e colocação da aceleração a 0
        v_t+1 = v_t * player_decay
        a_t+1 = 0
```

Algoritmo 2: Algoritmo do Modelo de Movimento dos Jogadores do Simulador

No caso de o jogador ser de um tipo heterogéneo, o valor de *player_accel_max*, *player_speed_max* ou de outros parâmetros do algoritmo, podem ser diferentes.

8.3.5.3 Movimento – Rotação

Enquanto o comando ***dash*** é utilizado para acelerar um jogador na direcção do seu corpo, o comando ***turn*** é utilizado para alterar a direcção do corpo do jogador. O argumento do comando corresponde ao *Momento* de rotação que pode ter valores compreendidos entre *[minmoment, maxmoment]*, actualmente definidos como –180 e 180 graus respectivamente:

(**turn** <*Momento*>)

Se o jogador se encontra parado no campo, o ângulo de rotação do jogador será igual ao *Momento* incluído no comando ***turn***. No entanto, de forma a tornar a simulação mais real, foi introduzida a noção de inércia. Desta forma, o ângulo de rotação do jogador não é igual ao momento mas sim dependente deste e da velocidade de movimento do jogador. Quando um jogador se desloca com maior velocidade, é mais difícil realizar a rotação devido ao efeito de inércia:

$$\hat{A}ngulo\_Real = Momento * (1.0 + Ruído) / (1.0 + inertia\_moment * |v_t|)$$

$$Ruído = Random\ (-player\_rand, player\_rand)$$

[57] Desprezando os efeitos do ruído e do vento (caso esteja activado).

Onde *inertia_moment* é um parâmetro do simulador (actualmente com o valor 5.0). *Ruído* é um valor aleatório retirado de uma distribuição uniforme entre *[-player_rand, player_rand]*. $|v_t|$ corresponde ao valor absoluto da velocidade do jogador. Como um jogador só pode executar um comando por ciclo, ou seja não pode realizar um ***dash*** e um ***turn*** simultaneamente no mesmo ciclo, a sua velocidade máxima quando executar um ***turn*** será de *player_speed_max * player_decay*, o que com os parâmetros actuais do servidor, corresponde a um valor de 0.4. Desta forma, o máximo que um jogador pode rodar a esta velocidade será de $\pm 180*1.0/(1.0+5.0*0.4) = \pm 60$[58].

As complexidades introduzidas no modelo de movimento dos jogadores, tais como a adição de ruído, energia limitada, inércia, etc., tornam problemas que à partida poderiam parecer de muito simples resolução, tais como a definição de trajectórias de movimento ou o planeamento de intercepções de bola, em problemas extremamente complexos de resolver.

8.3.5.4 Controlo da Bola – Chuto (*Kick*)

A segunda classe de comandos básicos está relacionada com a interacção dos agentes com a bola. Actualmente[59], existem três comandos deste tipo:

(**kick** <*Potência*> <*Direcção*>)

(**catch** <*Direcção*>)

(**tackle** <*Potência*>)

O comando ***kick*** serve para impulsionar a bola com uma determinada *Potência*, numa dada *Direcção*. O Comando ***catch*** só pode ser executado pelo guarda-redes e serve para o mesmo se "lançar para o chão" numa dada *Direcção* e agarrar a bola. O comando ***tackle*** serve para um jogador se "lançar para o chão" e chutar a bola com uma dada *Potência* na direcção do seu corpo. É útil para desarmar adversários que tenham a bola em sua posse.

Quando um jogador emite um comando ***kick***, deve fornecer dois parâmetros: a *Potência* do chuto, que vai determinar a aceleração dada à bola e que deve ter um valor entre *[minpower, maxpower]* e a *Direcção* do chuto que deve ser dada em graus e estar contida entre *[minmoment, maxmoment]*. Quando o comando ***kick*** chega ao servidor, só será executado se a bola estiver a uma distância do jogador inferior à máxima distância de chuto que é definida por *ball_size + player_size + kickable_margin*, ou seja, se a distância entre o exterior da bola e o exterior do jogador for inferior a *kickable_margin*. Jogadores heterogéneos podem ter diferentes capacidades relativamente aos parâmetros de chuto, nomeadamente diferentes *kickable_margin*.

---

[58] Desprezando os efeitos do ruído.

[59] Nas regras oficiais das competições em 2002 – Versão 8.x do simulador.

A eficiência com que o comando é executado depende da posição relativa da bola ao jogador. Se a bola estiver próxima e à frente do jogador, o comando será executado com maior eficiência do que se a bola estiver mais afastada e atrás do jogador:

*PotFinal = Potência * kick_power_rate **
*(1 - 0.25*Dir_Bola/180 – 0.25*Dist_Bola/kickable_margin)*

Em que *Dir_Bola* é a direcção da bola relativamente ao corpo do jogador e *Dist_Bola* é a distância da bola relativamente ao corpo do jogador. Desta forma se a bola estiver atrás do jogador (*Dir_Bola = 180*) e afastada do mesmo (*Dist_Bola = kickable_margin*) a eficciência do chuto será unicamente 50%.

Existe também um parâmetro *kick_rand* que permite adicionar ruído ao chuto dos jogadores. Para os jogadores normais este parâmetro tem o valor 0, podendo no entanto ter valores distintos para jogadores heterogéneos.

```
// Para cada comando kick válido aplicado por um jogador acelerar a bola na
direcção correspondente ao comando kick e com a potência final
      PARA ComandoKick = 1 ATÉ NComandosKick FAÇA
            SE Valido(ComandoKick) ENTÃO
                  Bat = Bat + PotFinal * DirecçãoKick
// Para cada comando tackle válido e bem sucedido aplicado por um jogador
acelerar a bola na direcção ao corpo do jogador e com a potência correspondente
      PARA ComandoTackle = 1 ATÉ NComandosTackle FAÇA
            SE Valido(ComandoTackle) E Sucesso(ComandoTackle) ENTÃO
                  Bat = Bat + PotTackle * DirecçãoPlayer
// Limitação da aceleração da bola
      |Bat| = Max(|Bat|, ball_accel_max)
// Cálculo da velocidade da bola adicionando a aceleração
      Bvt = Bvt + Bat
// Limitação da velocidade da bola ao máximo admissível
      |vt| = Max(|vt|, ball_speed_max)
// Adição de Ruído à velocidade da bola
      Bvt = Bvt + RuídoBola(ball_rand)
// Cálculo da nova posição da bola
      Bpt+1 = Bpt + Bvt
// Redução da Velocidade da bola utilizando o "decay" e colocação da aceleração
a 0
      Bvt+1 = Bvt * ball_decay
      Bat+1 = 0
```

Algoritmo 3: Algoritmo do Modelo de Movimento da Bola

O efeito de um comando *kick* será acelerar a bola na direcção correspondente ao ***kick***. O algoritmo 3 demonstra (de forma simplificada) o processo utilizado no simulador para calcular a velocidade e posição da bola em cada ciclo, tendo em conta os comandos ***kick*** e ***tackle*** emitidos pelos diversos jogadores.

Parâmetro	*Valor*
Minpower	*-100*
maxpower	*100*
minmoment	*-180*
Maxmoment	*180*
kick_power_rate	*0.027*
kick_rand	*0*

Parâmetro	*Valor*
ball_size	*0.085*
player_size	*0.3*
kickable_margin	*0.7*
ball_accel_max	*2.7*
ball_speed_max	*2.7*
ball_decay	*0.94*
ball_rand	*0.05*

Tabela 16: Parâmetros que afectam a acção kick (e valores para competições em 2002).

Embora o modelo de chuto da bola não tenha sofrido alterações nos últimos anos, devido à alteração dos parâmetros (aumento de *kick_power_rate* de 0.016 para 0.027), o problema do chuto da bola foi simplificado e, em muitos casos, não é agora necessário múltiplos ***kicks*** para acelerar a bola para a sua velocidade máxima. Na tabela 16 são apresentados os parâmetros actuais.

8.3.5.5 Controlo da Bola – Agarrar a Bola (*Catch*)

O guarda-redes é o único agente de cada equipa que tem capacidade para executar um comando de ***catch*** de forma a agarrar a bola. O comando ***catch*** toma como único parâmetro a *Direcção* que tem como valores válidos o intervalo *[minmoment, maxmoment]*. Quando o comando ***catch*** chega ao servidor, só será executado se as seguintes pré-condições forem obedecidas: o agente que enviou o comando ser do tipo guarda-redes; o modo de jogo ser *play_on*; a bola estar dentro da grande-área; a bola estar na área coberta pelo ***catch***, ou seja, num rectângulo com comprimento *catchable_area_l* e largura *catchable_area_w* na *Direcção* em que foi efectuado o ***catch*** (figura 58). Neste caso, o guarda-redes terá uma probabilidade de *catch_probability* de agarrar a bola.

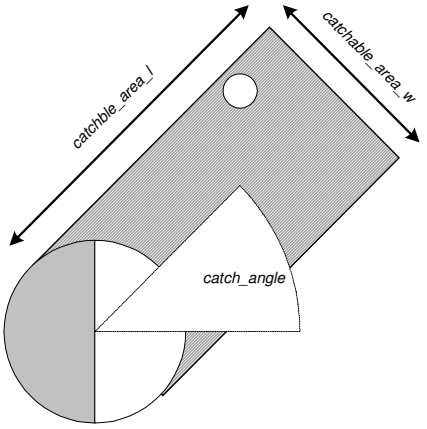

Figura 58: Modelo de Catch do Simulador

Se o comando de ***catch*** não for bem sucedido, demora *catch_ban_cycle* ciclos até que o guarda-redes possa enviar um novo comando de ***catch***[60]. No caso do comando ser bem sucedido, o modo de jogo muda imediatamente para *goalie_catch_ball_[l|r]* e logo após, muda para *free_kick[l|r]*. O guarda-redes pode então utilizar o comando ***move*** para se reposicionar na área e recolocar a bola em jogo. Os parâmetros do servidor importantes para a acção de ***catch*** encontram-se detalhados na tabela 17.

Parâmetro	***Valor***
Minmoment	*-180*
Maxmoment	*180*
catchable_area_l	*2.0*
catchable_area_w	*1.0*

Parâmetro	***Valor***
catch_probability	*1.0*
catch_ban_cycle	*5*
goalie_max_moves	*2*

Tabela 17: Parâmetros que afectam a acção catch.

8.3.5.6 Controlo da Bola - *Tackle*

A introdução do novo comando de ***tackle***, efectuada em 2002[61], tem como principal objectivo aumentar o realismo providenciando uma forma realista de desarmar jogadores adversários que se encontram na posse da bola. O formato do comando é:

(**tackle** <*Potência*>)

O efeito do tackle corresponde ao chuto da bola na direcção em que se encontra virado com a potência especificada no comando. O sucesso do ***tackle*** não é no entanto garantido sendo determinado probabilisticamente baseado na posição da bola relativamente ao jogador que efectua o ***tackle***. Quanto mais próxima estiver a bola, maior será a probabilidade de sucesso do ***tackle***. Se a bola estiver a mais de 2 metros de distância segundo o eixo dos xx ou 1 metro de distância segundo o eixo dos yy do jogador, esta probabilidade será nula

Após tentar efectuar um ***tackle*** um jogador fica imóvel durante 10 ciclos. A informação visual contém uma indicação se um dado jogador (visível) efectuar um ***tackle***. Um jogador que efectuar um ***tackle*** recebe informação sobre o número de ciclos que ainda tem de esperar até poder efectuar um novo comando.

8.3.5.7 Controlo da Percepção – Rotação do Pescoço Flexível

Utilizando o comando ***turn_neck***, os jogadores podem rodar o seu pescoço flexível, obtendo um importante controlo sobre a percepção. A sintaxe do comando é:

---

[60] Pode no entanto, neste intervalo, enviar comandos de kick para chutar a bola.

[61] O comando tackle encontra-se disponível a partir do servidor 8.x

(**turn_neck** <*Ângulo*>)

O comando ***turn_neck*** modifica a direcção do pescoço do jogador e consequentemente do seu ângulo de visão relativamente ao seu corpo de *Ângulo* graus. O pescoço pode ser rodado entre *[minneckmoment, maxneckmoment]*, correspondendo a *[-180, 180]* graus na versão actual do simulador, relativamente ao corpo do jogador. No entanto, o valor final da direcção do pescoço relativamente ao corpo tem de assumir um valor compreendido entre *[minneckang, maxneckang]*, correspondendo a *[-90, 90]* graus. Quando um comando ***turn_neck*** conduz a um valor ilegal deste ângulo, o comando é adaptado de forma a que a direcção do pescoço relativamente ao corpo, permaneça dentro dos limites. No controlo da direcção de visão utilizando este comando é necessário ter em conta que o ângulo de visão varia também com o comando ***turn***, mesmo que nenhum comando ***turn_neck*** seja executado.

Contrariamente ao comando ***turn***, não é adicionado ruído ao comando ***turn_neck*** nem a eficiência deste depende da velocidade do jogador. Desta forma, um comando *(**turn_neck** 80)* conduzirá a que o pescoço do jogador rode exactamente de 80 graus no sentido positivo relativamente ao corpo do jogador, independentemente da velocidade a que o jogador se desloque.

8.3.5.8 Controlo da Percepção – Configuração da Visão

A configuração da percepção visual de um agente pode ser efectuada utilizando o comando ***change_view*** com a seguinte sintaxe:

(**change_view** <*Abertura*> <*Qualidade*>)

Este comando toma dois parâmetros: *Abertura* do cone de visão (*narrow, normal* ou *wide*) e a *Qualidade* da informação visual (*high* ou *low*). Os jogadores podem utilizar o comando ***change_view*** de forma a trocarem a *Abertura* ou *Qualidade* da informação visual por uma maior frequência do envio da mesma por parte do servidor. Uma maior qualidade visual e um cone de visão mais largo conduzem a informação visual menos frequente.

8.3.5.9 Controlo da Percepção – Configuração da Audição

A introdução em 2002 do comando ***attentionto*** introduziu novos desafios no controle da percepção dos agentes da liga de simulação. Utilizando este comando é agora possível efectuar uma escolha selectiva de quais os agentes que são prioritários ouvir em cada instante. A sintaxe do comando é:

(**attentionto** <*Equipa*> <*Unum*>) | (**attentionto** *off*)

onde <*Equipa*> *::=* *opp* | *our* | *l* | *r* | *left* | *right* | <*Nome_Equipa* > permite indicar qual a equipa a que pertence o jogador e <*Unum*> é um inteiro que permite especificar o número desse jogador. Cada comando ***attentionto*** sobrepõe-se a um anterior.

A primeira forma do comando especifica o jogador ao qual deve ser dada atenção. As mensagens deste jogador deverão ser ouvidas preferencialmente às dos outros jogadores. Se esse jogador enviar várias mensagens no mesmo ciclo, uma é escolhida ao acaso. No caso de esse jogador não enviar qualquer mensagem, uma mensagem de um outro jogador é escolhida aleatoriamente.

A segunda forma do comando permite desligar a atenção dada a um determinado jogador de forma a que as mensagens a escutar sejam escolhidas aleatoriamente de qualquer jogador.

8.3.5.10 Comunicação - Say

O modelo de comunicação do *soccerserver* baseia-se na possibilidade de envio de mensagens para o ambiente, que são distribuídas pelos agentes que estão próximos. Este modelo de comunicação sofreu em 2002 a introdução de uma série de restrições à comunicação auditiva e a adição de um novo comando que permite a um agente apontar numa determinada direcção.

Sempre que um agente deseje enviar uma mensagem aos restantes jogadores, poderá emitir um comando say com a forma:

(**say** <*Messagem*>)

O parâmetro único deste comando *Messagem* corresponde à mensagem que o jogador deseja emitir. O tamanho desta mensagem está limitado pelo parâmetro *say_msg_size* (correspondente a 512 bytes até 2001 mas reduzido para apenas 10 bytes em 2002). Os caracteres da mensagem têm de provir do conjunto [0..9a..zA..Z ().+-*/?_<>] (sem os parênteses), o que implica a disponibilidade de 74 símbolos unicamente para efectuar a comunicação. A explicação detalhada dos parâmetros de comunicação e respectivos significados e alterações ao longo do tempo foi já realizada na secção 8.3.4.2.

8.3.5.11 Comunicação - Pointto

Desde 2002 os jogadores têm também a capacidade de comunicar visualmente com os restantes jogadores apontando para pontos específicos do campo. A sintaxe deste novo comando é:

(**pointto** <*Distância*> <*Direcção*>) | (**pointto** *off*)

Um jogador após apontar para uma dada direcção permanecerá durante um mínimo de 5 ciclos e um máximo de 20 ciclos ou outro comando pointto emitido, a apontar nessa direcção. Os jogadores permanecerão a apontar para a mesma direcção independentemente do seu movimento no campo.

Os restantes jogadores recebem na sua informação visual a direcção em que um dado jogador está a apontar, na forma:

(p <Equipa> <Jogador>) <Distância> <Direcção> <DistVar> <DirVar> <DirCorpo> <DirCabeça> <DirAponta> ou

(p <Equipa>) <Distância> <Direcção> <DirAponta>

Ruído dependente da distância a que um jogador se encontra do jogador que está a apontar é adicionado.

8.3.6 Agentes Heterogéneos

Os agentes heterogéneos foram introduzidos em 2001 como forma de estimular a investigação realizada em diversos domínios, com ênfase para a alocação dinâmica de recursos. No início de cada jogo são sorteados aleatoriamente seis tipos de agentes heterogéneos que possuem características distintas dos jogadores normais. Estas características incluem, entre outras, velocidade, aceleração, recuperação de energia e momento de inércia distintos. A tabela 18 apresenta um resumo dos parâmetros que podem sofrer alterações nos jogadores heterogéneos.

Parâmetro	***Valor***	***Alcance***	***Descrição***
player_speed_max	*1.0*	*[1.0, 1.2]*	*Jogador com maior velocidade máxima*
stamina_inc_max	*45.0*	*[25.0, 45.0]*	*Jogador que recupera menos energia em cada ciclo*
player_decay	*0.4*	*[0.4, 0.6]*	*Jogador cuja velocidade decresce mais lentamente*
inertia_moment	*5.0*	*[5.0, 10.0]*	*Jogador que roda com mais dificuldade em movimento*
dash_power_rate	*0.006*	*[0.006, 0.008]*	*Jogador que acelera mais rapidamente*
player_size	*0.3*	*[0.1, 0.3]*	*Jogador com um corpo mais pequeno*
kickable_margin	*0.7*	*[0.7, 0.9]*	*Jogador com maior área de chuto*
kick_rand	*0.0*	*[0.0, 0.1]*	*Jogador com menor precisão de chuto*
extra_stamina	*0.0*	*[0.0, 100.0]*	*Jogador com alguma energia extra*
effort_max	*1.0*	*[0.8, 1.0]*	*Jogador que entra mais dificilmente em esforço*
effort_min	*0.6*	*[0.4, 0.6]*	*Jogador que sai mais facilmente de esforço*

Tabela 18: Parâmetros dos Jogadores Heterogéneos

Estes parâmetros têm sido utilizados como base para a definição de *"tradeoffs"* em cada competição de forma a garantir que embora os jogadores heterogéneos possuam algumas características físicas melhores que os jogadores normais, possuem também no entanto algumas características piores. Exemplos de *"tradeoffs"* utilizados em competições incluem a troca de aceleração por recuperação de energia (correspondendo a um jogador que acelera rapidamente mas também se cansa rapidamente) ou a troca de área de chuto por precisão de chuto (correspondendo a um jogador com pernas longas capaz de cortar bolas aos adversários com maior facilidade mas que têm dificuldades na realização de passe precisos).

No início do jogo, o treinador de cada equipa é informado de quais os tipos de jogadores heterogéneos que dispõe para o jogo. Pode então substituir os jogadores normais por jogadores de tipos heterogéneos até a um máximo de 3 de cada tipo. No decurso do jogo, o treinador dispõe de três substituições que poderá utilizar para refrescar a equipa (os

jogadores que entram possuem a energia e capacidade de recuperação ao máximo) e alterar os tipos de jogadores (introduzindo jogadores heterogéneos de tipos distintos).

8.3.7 O Treinador

O treinador é um agente privilegiado que pode ser utilizado por uma equipa para auxiliar os seus jogadores. O simulador disponibiliza dois tipos de treinadores: O treinador *on-line* e o treinador *off-line*. Ambos os treinadores recebem informação global e livre de erros relativa a todos os objectos presentes no campo. Para além disto, o treinador recebe ainda todas as mensagens enviadas pelos jogadores e pelo árbitro e pode enviar mensagens audíveis por todos os jogadores da sua equipa.

Diversas restrições são incluídas neste agente de forma a que não possa ser utilizado como um controlador centralizado da equipa. No entanto, é também neste cliente que se situam importantes capacidades para o controlo global da equipa tais como a capacidade para seleccionar os jogadores heterogéneos a utilizar e efectuar substituições durante o jogo (de jogadores cansados ou de forma a efectuar a troca do tipo de jogadores).

8.3.7.1 Capacidades do Treinador *off-line*

A diferença essencial entre o treinador *off-line* e o treinador *on-line* é que o primeiro não pode ser utilizado durante jogos oficiais, servindo exclusivamente para auxiliar na fase de desenvolvimento da equipa. O treinador *off-line* pode exercer um muito maior controlo sobre a equipa relativamente ao treinador *on-line*. Ele pode controlar o modo de jogo desactivando o árbitro artificial se necessário. Pode ainda movimentar os jogadores de ambas as equipas e a bola para qualquer ponto do campo. Pode também atribuir velocidades e orientações aos jogadores o que se revela muito útil para ensaiar determinadas situações durante os treinos. Com o auxílio deste agente é muito simples criar situações de treino semelhantes às que os treinadores reais de futebol criam durante os treinos de equipas reais de futebol.

8.3.7.2 Capacidades do Treinador on-line

O treinador *on-line* é utilizado durante os jogos para fornecer informação adicional e enviar instruções aos jogadores. As suas capacidades são muito limitadas pois não pode controlar o modo de jogo ou os jogadores. Adicionalmente, para que ele não possa ser utilizado como um controlador centralizado, as suas capacidades de comunicação com os jogadores são fortemente restringidas. O treinador só pode comunicar com os seus jogadores quando o jogo está parado ou no decurso normal do jogo, mas utilizando mensagens codificadas numa linguagem especial e que chegam aos jogadores com atraso relativamente ao envio.

Embora as restrições às capacidades do treinador sejam severas, dado que este agente recebe informação global e livre de erros e não tem solicitações em tempo-real, pode gastar o seu tempo realizando análises mais complexas acerca do comportamento do adversário e da estratégia, tácticas e formações a utilizar pela sua equipa. Constitui desta forma uma ferramenta valiosa para analisar as forças e fraquezas do oponente, analisar o decurso do jogo e decidir quais os conselhos tácticos a fornecer aos seus jogadores.

8.3.7.3 Linguagem "Standard" de Comunicação

De forma a permitir a um treinador artificial treinar uma equipa de futebol robótico simulado é necessário definir uma linguagem de comunicação apropriada. Esta linguagem será tanto mais apropriada quanto genérica e de compreensão geral for. Desta forma, uma nova área de investigação surgiu na comunidade RoboCup com o objectivo de criar uma linguagem standard que permita a um dado treinador artificial ser capaz de treinar diferentes equipas de futebol robótico simulado.

A primeira linguagem criada neste sentido foi COACH UNILANG [Reis e Lau, 2002] definida como uma linguagem standard de alto-nível para treinar uma equipa de futebol (robótico). COACH UNILANG é uma linguagem baseada em conceitos comuns do futebol real e robótico, tais como: regiões, períodos de tempo, situações, tácticas, formações, tipos e comportamentos de jogadores. Inclui construtores que permitem definir tácticas genéricas de futebol de uma forma directa e muito simples. Uma descrição mais detalhada da linguagem pode ser encontrada no capítulo 9 deste trabalho ou em [Reis e Lau 2002].

A especificação da linguagem COACH UNILANG foi tornada pública em Janeiro de 2001 e recolheu um amplo apoio da comunidade do RoboCup. No entanto a federação RoboCup optou por utilizar como linguagem *"standard"* no simulador oficial uma linguagem de baixo-nível designada CLang, importando unicamente alguns conceitos básicos da linguagem COACH UNILANG. Esta linguagem *"standard"* foi utilizada como base para uma competição de treinadores no RoboCup 2001 em Seattle. No entanto, unicamente 4 equipas participaram nesta competição. Adicionalmente, foi concluído pela maioria dos participantes e por grande parte dos assistentes da competição que a linguagem standard não era suficientemente expressiva e era demasiado baixo-nível para treinar uma boa equipa de futebol robótico simulado. Desta forma, nenhum dos participantes utilizou com sucesso esta linguagem para treinar a sua própria equipa na competição oficial e adicionalmente foi concluído que na competição de treinadores, as quatro equipas jogavam melhor sem treinador do que com um treinador comunicando em CLang.

Estas primeiras experiências levaram à conclusão da necessidade de uma linguagem de alto-nível mais preocupada com o nível da equipa, do que com o comportamento individual dos jogadores. Bons jogadores artificiais de futebol simulado já possuem

capacidades de decisão individual apuradas e necessitam sobretudo de orientação sobre como funcionarem como elementos de uma equipa. No entanto, para o ano de 2002, contra a opinião da comunidade, o comité técnico decidiu-se pela manutenção da linguagem Clang como base da competição de treinadores, importando no entanto para esta linguagem uma parcela significativa dos conceitos introduzidos na linguagem COACH UNILANG[62] [Reis e Lau, 2002].

8.4 Abordagens mais significativas ao Futebol Robótico Simulado

Uma das principais motivações das competições RoboCup consiste na disponibilização de um problema standard composto por inúmeros subproblemas cuja resolução pode ser efectuada através da utilização de diferentes metodologias. Desta forma, diferentes equipas utilizam metodologias distintas para a resolução do mesmo problema. Por exemplo, no problema de chutar a bola, algumas equipas utilizam abordagens analíticas [Stone et al., 1999], outras utilizam aprendizagem por reforço [Riedmiller et al., 2000] e outras metodologias de optimização *on-line* [Reis e Lau, 2001a].

As principais metodologias desenvolvidas pelas principais equipas do RoboCup ao longo dos últimos anos foram [Reis, 2001a] [Reis, 2001b]:

- **Formações e Papéis.** Metodologias que permitem definir formações espaciais e atribuir aos jogadores papéis e eventualmente formas de efectuar a troca desses papéis;
- **Estratégias e tácticas.** Metodologias que permitem definir a alto-nível a estratégia para um dado jogo e alterar o comportamento da equipa globalmente durante esse jogo.
- **Modelização de Oponentes.** Criação de modelos da equipa oponente e dos seus jogadores de forma a poder adequar o comportamento da equipa a esse oponente.
- **Metodologias de aprendizagem.** Utilização de metodologias de aprendizagem tais como a aprendizagem por reforço de forma a desenvolver *low-level skills* (chuto, drible, intercepção, etc.) ou métodos de decisão individual (passar, chutar, driblar) ou mesmo de decisão colectiva.
- **Técnicas de Optimização.** Aplicação de metodologias de optimização para o desenvolvimento de *low-level skills* para os jogadores.

---

[62] A equipa FC Portugal embora tenha manifestado a sua oposição ao formato e linguagem utilizada na competição, participou e venceu a mesma. Na competição ficou mais uma vez provado que a linguagem CLang tem um efeito detrimental nas equipas treinadas.

- **Planeamento Flexível.** Desenvolvimento de metodologias que permitam criar, durante o jogo, planos flexíveis para a execução colectiva de determinadas tarefas durante o jogo.
- **Representação do conhecimento do futebol real.** Desenvolvimento de metodologias que permitam efectuar a recolha, representação e utilização de conhecimento recolhido através do contacto com especialistas da área (treinadores, jogadores e jornalistas).
- **Utilização do treinador *on-line*.** Definição de metodologias que permitam ao treinador analisar o jogo e decidir, em cada instante, qual a melhor táctica para a sua equipa.
- **Construção de Ferramentas de Análise e Debug.** Construção de ferramentas que permitam analisar jogos de futebol robótico simulado *on-line* ou *off-line* e que permitam analisar a informação disponível aos agentes e os seus processos de raciocício ao longo dos jogos.

8.5 Trabalho Relacionado e Aplicações do RoboCup

Existem diversas aplicações interessantes das tecnologias desenvolvidas no contexto da liga de simulação do RoboCup. Algumas delas encontram-se já em progresso, enquanto outras possuem um grande potencial para serem realizadas num futuro próximo. De entre estas aplicações destacam-se [Reis, 2001a] [Reis, 2002g]:

- *Routing* de pacotes de rede;
- Coordenação de pilotos de helicópteros sintéticos e análise de cenários de guerra;
- Simulação de combate aéreo e de Veículos (aéreos) inteligentes e autónomos;
- Coordenação automática de pessoas;
- Combate a catástrofes tais como incêndios florestais ou no âmbito do *RoboCup Rescue*;
- Realidade virtual;
- Educação;
- Análise de jogos de futebol;
- Coordenação de *AGVs* (*Automated Guided Vehicles*);
- Robótica industrial e controlo de processos industriais;
- Controlo de satélites;

- Limpeza de minas e de lixo radioactivo;
- Controlo de robôs hospitalares;
- Controlo Inteligente de Câmaras.

Stone et al. [Stone et al., 2000] na Carnegie Mellon University, aplicaram com sucesso diversas metodologias desenvolvidas no âmbito da liga de simulação do RoboCup ao domínio do *routing* de pacotes de rede. De entre estas metodologias destacam-se o *TPOT-RL – Team Partitioned, Opaque Transition Reinforcement Learning* [Stone et al., 2000], a aprendizagem máquina hierárquica em camadas (*Layered Learning*) [Stone et al., 2000], as formações adaptativas [Stone et al., 1999], a comunicação em canal único com reduzida largura de banda [Stone et al. 1999], os *Predictive, Locally Optimal Skills (PLOS)* e o posicionamento utilizando atracções e repulsões (SPAR) [Stone, 1998]. Embora os resultados publicados por Stone et al. na aplicação a este domínio não sejam tão extensos como os de aplicação ao futebol robótico, os resultados demonstram que muitas das metodologias desenvolvidas no RoboCup podem ser generalizadas para outros domínios[63].

Milind Tambe e Gal Kaminka na University of South Califórnia partiram de uma perspectiva inversa, aplicando ao RoboCup simulado metodologias desenvolvidas originalmente para outros domínios. O modelo de trabalho em equipa STEAM foi desenvolvido originalmente para a coordenação de pilotos de helicópteros em situações militares complexas e de larga escala [Tambe, 1997]. O RoboCup foi um domínio adicional onde esta metodologia foi demonstrada sem que, no entanto, tenha obtido resultados muito significativos. A mesma tecnologia formou a base do TEAMCORE, uma arquitectura de integração de aplicações distribuídas que foi demonstrada em diversos outros domínios tais como a coordenação automática de humanos [Tambe, 2000]. Gal Kaminka desenvolveu também, na mesma Universidade, um trabalho focalizado na monitorização e diagnóstico de equipas, que foi avaliado em três domínios distintos: futebol simulado, simulação militar e coordenação de humanos [Kaminka, 2000].

No domínio da detecção distribuída de incêndios florestais destaca-se o trabalho desenvolvido por Mikhail Prokopenko no CSIRO, Austrália [Prokopenko, 2002].

Uma aplicação relevante das metodologias desenvolvidas no âmbito do RoboCup, consiste na análise de jogos de futebol real. Diversos sistemas comerciais, tais como *Match Analysis* [MatchAnalysis, 2002] existem exactamente com a função de analisar jogos de futebol, providenciando estatísticas de alto-nível sobre o jogo para treinadores e profissionais da comunicação social. As equipas técnicas de uma percentagem muito significativa das principais equipas de futebol mundiais utilizam sistemas de análise automática de jogos. As metodologias desenvolvidas no âmbito da liga de simulação do

---

[63] O trabalho de Stone na sua tese de doutoramento e a sua aplicação nesta área, levou a que este cientista fosse contratado pela AT&T Research.

RoboCup, nomeadamente no que diz respeito ao cálculo automático de estatísticas de jogo, analisadores de jogo e comentadores automáticos têm, num futuro muito próximo, aplicação directa no domínio do futebol real. No âmbito do projecto FC Portugal, baseado no contacto com profissionais da área do futebol (jogadores, treinadores e jornalistas) foi desenvolvido um sistema de análise estatística de jogos de futebol, uma metodologia de definição de uma estratégia para um jogo de futebol e uma linguagem standard para comunicação treinador – equipa. Encontra-se actualmente em estudo a aplicação destas metodologias ao domínio do futebol real em Portugal.

O *RoboCup Rescue* é um dos domínios mais promissores para aplicação das metodologias desenvolvidas no âmbito do RoboCup. Esta nova iniciativa da Federação RoboCup está preocupada com o planeamento e execução de operações de resgate e salvamento em desastres de larga escala tais como terramotos e incêndios.

A robótica Industrial é outra aplicação futura do RoboCup que assume grande importância. O envio de robôs autónomos e inteligentes para situações perigosas diminui o número de vidas humanas colocadas em risco. Tarefas complexas em que as tecnologias principais desenvolvidas no RoboCup podem ser aplicáveis, incluem a exploração de terreno, gestão de armas nucleares, exploração lunar, limpeza de lixo radioactivo e a limpeza de minas.

8.6 O RoboCup como Competição Científica

As competições científicas têm em geral um grande potencial para acelerar o progresso científico num dado domínio. No entanto, a realização de competições como o RoboCup tem também diversos perigos para o avanço da ciência.

Os principais perigos das competições científicas [Stone, 1998] estão centrados na obsessão por vencer, com o desenvolvimento de soluções dependentes do domínio e com o custo do equipamento necessário para ter uma equipa competitiva.

Especialmente quando prémios monetários estão envolvidos, existe um certo incentivo a manter secretas as técnicas "vencedoras" de ano para ano, sem que o seu desenvolvimento permita avançar a ciência em geral. No RoboCup, a atribuição do "*scientific challange award*" procura diminuir o risco do secretismo, incentivando e premiando a publicação de trabalho científico de qualidade. Por outro lado, em diversas ligas, nomeadamente na liga de simulação, a disponibilização do binário de anos anteriores é condição necessária para a admissão à competição no ano seguinte. A disponibilização de código fonte é também largamente incentivada. Os campeões da liga de simulação de diversos anos (CMUnited, FC Portugal e Tsinghuaeolus) e outras equipas bem sucedidas (tais como os Karlsruhe Brainstormers, UvaTrilearn ou YowAI) têm disponibilizado partes ou mesmo a totalidade do seu código fonte na Internet.

O desenvolvimento de soluções dependentes do domínio é outro perigo das competições, em particular em domínios complexos como o RoboCup. Alguns autores consideram [Stone, 1999] que para produzir uma equipa de futebol robótico bem sucedida, a investigação em metodologias de coordenação e cooperação não é suficiente. No entanto, é convicção do autor que a investigação que leva realmente a obter bons resultados no RoboCup simulado situa-se a um nível elevado. Os detalhes de baixo nível muito ligados ao domínio, tais como desenvolver bons algoritmos de intercepção de bola, chuto, drible, etc. são de muito menor importância do que o desenvolvimento de algoritmos de coordenação, tais como alocação e troca de papéis, posicionamento estratégico, formalizações do conceito de estratégia, formas de definir e implementar tácticas e formações, etc.

O custo para ter uma equipa competitiva é muito relevante sobretudo nas ligas robóticas. A utilização de tecnologias caras já existentes pode conduzir a um grande benefício na competição sem que conduza a um avanço da ciência. Uma solução que a federação RoboCup tem utilizado nas ligas robóticas consiste em impor restrições às dimensões, sensores e actuadores dos robôs, tais como limites na potência de chuto ou na capacidade de controlo de bola.

Peter Stone afirma que um perigo da competição RoboCup está relacionado com a dificuldade em entrar nela, pois as equipas antigas têm uma considerável vantagem [Stone, 1999]. No entanto, ao longo dos últimos anos esta conclusão tem-se demonstrado errada no âmbito do RoboCup. Em 2000 e 2001, a liga de simulação foi vencida por equipas que se estrearam na competição (FC Portugal e Tsinghuaeolus). Factores que contribuem para este facto são a disponibilidade de código fonte (de diversas fontes) na Internet e as alterações às regras introduzidas de ano para ano pelo comité técnico da simulação. A equipa FC Portugal que venceu o RoboCup 2000 utilizou parte do código fonte (contendo os *low-level skills* e comunicação com o servidor) disponibilizado pelo campeão anterior (CMUnited99). A equipa Tsinghuaeolus que venceu o campeonato do mundo em 2001, utilizou o código fonte disponibilizado pelo FC Portugal em 2000 (contendo a definição de tácticas, formações e posicionamento estratégico).

Um outro perigo da competição consiste numa avaliação incorrecta dos resultados. Evidentemente que um único jogo entre duas equipas, não é uma experiência que permita retirar conclusões científicas válidas. Não se pode concluir que se a equipa *X* venceu a equipa *Y*, então todas as metodologias utilizadas pela equipa *X* serão mais bem sucedidas do que as utilizadas pela equipa *Y* [Stone, 1998]. Para além disso, o resultado do jogo é um dos muito factores a ser considerado numa perspectiva científica de análise. Factores como o tempo de posse de bola por região do campo, número de remates à baliza, cruzamentos, passes bem sucedidos, defesas do guarda-redes, etc. são muito relevantes na análise do jogo. Experiências controladas no laboratório têm de ser efectuadas para validar os resultados de investigação e as contribuições de investigação de cada equipa [Stone, 1999].

Embora existam diversos perigos na realização de competições científicas, os benefícios de as realizar superam largamente os problemas que advêm dessa realização. Os benefícios principais, no caso da liga de simulação, situam-se ao nível da existência de uma plataforma comum que permita comparar soluções, trocar ideias, aferir do progresso realizado e realizar competições com a natural motivação dada pela vontade de vencer. A competição em si própria, constitui uma forte inspiração para a investigação. A vontade de vencer, criando para tal, uma equipa competitiva e como tal criando soluções inovadoras para os diversos problemas, que é necessário resolver para criar essa equipa, tem constituído ao longo dos anos um dos motores para o progresso científico no âmbito do RoboCup.

A realização de competições RoboCup permite juntar um vasto conjunto de investigadores que realizam investigação em diversos domínios relacionados com Sistemas Multi-Agente e que utilizam o futebol robótico como domínio de aplicação. As equipas têm de resolver os mesmos problemas no mesmo domínio e utilizam para tal metodologias distintas. As diferentes soluções podem, através da competição, ser comparadas de forma directa. Como a competição RoboCup é realizada todos os anos e os binários das melhores equipas do ano anterior são disponibilizados, verifica-se que todas as equipas antes de participarem no campeonato procuram desenvolver metodologias que lhes permitam vencer as melhores equipas do ano anterior. Esta motivação de vencer os melhores contribui decisivamente para o progresso da investigação na área.

A existência do *soccerserver* e a disponibilização de uma enorme quantidade de equipas na Internet, permite ainda aos investigadores realizarem testes no seu laboratório e terem uma noção imediata do seu progresso. Por exemplo, de forma a testar a utilidade de uma determinada metodologia é possível realizar testes no laboratório jogando contra uma outra equipa, seleccionada de entre a grande variedade de equipas disponibilizadas na Internet, activando ou não a técnica em estudo.

O entusiasmo que o RoboCup consegue induzir aos professores e aos estudantes a todos os níveis é outro benefício evidente. É mais fácil envolver estudantes em investigação séria utilizando uma aplicação motivadora como o RoboCup [Stone, 1999]. Usualmente as equipas incluem desde professores até estudantes de pós-graduação e graduação, todos trabalhando em conjunto. Adicionalmente, as competições são muito úteis para leccionar cursos da área. Diversos cursos quer ao nível da graduação, quer ao nível de pós-graduação, utilizam o simulador do RoboCup como base para aplicação das metodologias e usualmente incluem uma mini-competição entre os estudantes.

Outro benefício da realização de competições RoboCup consiste no facto de impor prazos bem definidos para a criação de um Sistema Multi-Agente totalmente funcional. Usualmente, o desenvolvimento de agentes realmente funcionais é muitas vezes desprezado quando se realiza investigação científica. No entanto, para competir no RoboCup não basta colocar um ou dois módulos dos agentes em funcionamento e escrever

um artigo explicando como esses módulos um dia irão funcionar. É necessário desenvolver agentes completos que realmente funcionem do ponto de vista individual e como componentes do Sistema Multi-Agente.

8.7 Conclusões

O RoboCup veio estimular consideravelmente a investigação realizada em Inteligência Artificial Distribuída e Robótica Inteligente. Mas mais do que isso, veio chamar a atenção da comunicação social e do público em geral para esta investigação. As suas ligas colocam problemas de investigação distintos mas ao mesmo tempo interrelacionados. A liga de simulação analisada com maior detalhe neste trabalho, inclui complexidades encontradas nos sistemas robóticos (tais como erros na percepção e acção) mas ao mesmo tempo utiliza um cenário realista do ponto de vista da tarefa cooperativa a executar pelas equipas de agentes – jogar um desafio de futebol.

A liga de simulação coloca um vasto conjunto de desafios aos investigadores da área da Inteligência Artificial e Sistemas Multi-Agente. As características do domínio e desafios associados mais importantes colocados pelo simulador incluem [Reis, 2001a]:

- **Simulação em tempo-real.** O simulador actualiza o mundo em intervalos de tempo discretos (ciclos de simulação), cada qual com a duração de 100ms. Neste intervalo de tempo, os agentes recebem diversos tipos de informação sensorial e têm de enviar pedidos de execução de acções para o simulador. Isto requer aos agentes, capacidade de raciocínio em tempo-real.
- **Diversas fontes de Informação Sensorial e disponibilidade de sensores configuráveis.** Os agentes possuem três tipos de sensores: visual, auditivo e físico. O sensor visual fornece aos agentes informação (distâncias, direcções, etc.) sobre os objectos que se encontram no seu campo de visão. Funciona também como um sensor de proximidade fornecendo informação sobre objectos que estão muito próximos. A informação visual é bastante incompleta (os agentes possuem cones de visão reduzidos), é fornecida numa perspectiva relativa ao agente e com erros. No entanto, a disponibilidade de marcadores ao longo do campo permite ao agente converter esta informação em coordenadas absolutas. O sensor auditivo fornece aos agentes a informação relativa às mensagens emitidas pelos outros agentes, pelo árbitro e pelo treinador. Tem no entanto um alcance e capacidade muito limitados. O sensor físico fornece informação sobre a velocidade e energia do jogador e acções executadas pelo agente. Os sensores são configuráveis, permitindo a sua utilização inteligente por parte dos agentes. O sensor visual é configurável no que diz respeito à qualidade/frequência da informação e ângulo relativo ao corpo do agente. O sensor auditivo é configurável relativamente à prioridade atribuída às mensagens de diferentes agentes.

- **Modelo energético realista.** O simulador evita que os agentes corram continuamente no campo à máxima velocidade, atribuindo a cada jogar energia limitada. Sempre que o jogador executa um comando *dash*, gasta energia. No entanto, a energia é também restaurada em cada ciclo de acordo com a capacidade de recuperação do jogador. No caso de os jogadores permanecerem cansados durante muito tempo perdem capacidade de recuperação e eficiência de movimentos. Este modelo energético torna necessário que os agentes raciocinem em termos de consumo dos seus recursos energéticos, tendo em conta a importância das acções a executar.
- **Comunicação pouco fiável e com baixa largura de banda.** Ao contrário do que acontece na maioria das aplicações de SMA, neste domínio, os agentes não podem comunicar directamente uns com os outros. As comunicações são efectuadas através do simulador usando os protocolos *say* e *hear* que restringem a comunicação de diversas formas. Acresce a este facto que os 22 agentes utilizam o mesmo canal de comunicação que possui reduzida largura de banda e é muito pouco fiável.
- **Percepção e acção assíncronas.** Os agentes podem executar uma acção em cada 100ms. No entanto, a informação visual (com os parâmetros visuais por defeito), chega aos agentes de 150 em 150ms. Desta forma, e uma vez que é crucial aos agentes executarem uma acção em cada ciclo, torna-se necessário que os agentes sejam capazes de enviar comandos para o servidor mesmo sem receber informação sensorial.
- **Ambiente multi-objectivo, parcialmente cooperativo e parcialmente adverso.** Os agentes possuem dois objectivos de alto-nível contraditórios: defender para evitar sofrer golos e atacar para marcar golos. O ambiente é parcialmente cooperativo e os agentes têm de se comportar como membros de uma equipa. É também parcialmente adverso, no sentido que existe uma outra equipa de agentes exactamente com os objectivos opostos.
- **Vasto conjunto de acções de baixo-nível disponíveis.** Os agentes possuem acções de duas categorias: primárias (*kick, dash, turn, tackle, catch* e *move*) e secundárias (*turn_neck, change_view, attentionto, pointto* e *say*). Em cada ciclo, unicamente uma das acções primárias pode ser executada, embora diversas acções secundárias possam ser executadas concorrentemente. Se o agente enviar mais do que uma acção primária, o servidor escolhe aleatoriamente qual das acções enviadas vai executar, ignorando todas as outras. Não existe nenhuma garantia que uma dada acção seja executada pelo servidor, pelo que é necessário confirmar a execução das acções utilizando os sensores. Ruído é adicionado às acções de diversas formas.
- **Necessidade de transformar as acções de baixo-nível em comandos de alto-**

nível. De forma a executar uma determinada acção simples, tal como chutar a bola numa dada direcção, driblar com a bola ou interceptar uma bola em movimento, é necessário executar uma sequência de acções elementares de baixo-nível. Por exemplo: várias acções *kick* são habitualmente necessárias para chutar a bola numa dada direcção; vários *dash* e *turn* intercalados são necessários para interceptar a bola; vários *dash, turn* e *kick* são necessários para driblar com a bola numa dada direcção. No entanto, os erros contidos na percepção e acção, o ruído que o simulador introduz na movimentação dos objectos em campo e as acções imprevisíveis dos restantes agentes, implicam a necessidade de construir planos flexíveis para realizar tais acções. Torna-se necessário efectuar replaneamentos em cada ciclo, em face de nova informação sensorial disponível.

- **Necessidade de criar acções colectivas complexas.** A complexidade da tarefa cooperativa a executar, implica a necessidade de criar em tempo-real, acções colectivas complexas tais como passes, desmarcações, jogadas colectivas (implicando a troca de bola entre vários jogadores), movimentos defensivos colectivos (implicando dois ou mais jogadores executarem acções defensivas em conjunto, tais como cortar a bola a um adversário ou marcarem um conjunto de avançados), definição de formações, etc.
- **Agentes heterogéneos.** A introdução de agentes heterogéneos em 2001 trouxe desafios adicionais para o domínio. Cada equipa pode escolher os seus jogadores de entre sete tipos gerados aleatoriamente antes do início do jogo. Os diferentes tipos de jogadores possuem capacidades distintas, baseadas num conjunto de *tradeoffs* sobre um número limitado de parâmetros. Torna-se necessário seleccionar os melhores tipos de jogadores e efectuar um mapeamento desses tipos aos agentes de acordo com as suas funções.
- **Agente treinador com visão global do mundo.** A disponibilidade de um agente privilegiado que possui uma visão global e sem erros do mundo, permite analisar as capacidades e fraquezas do oponente de uma forma muito mais segura. O treinador é uma ferramenta ideal para estudar a estratégia do adversário e as capacidades dos seus jogadores e, em face desta análise, decidir qual a táctica a utilizar pela sua equipa no decurso do jogo. O treinador é também responsável por seleccionar e substituir os jogadores heterogéneos. Tendo em conta que este agente só pode comunicar quando o jogo se encontra parado (ou a intervalos regulares, durante o jogo, mas com atraso adicionado às suas mensagens), torna-se necessário que a informação comunicada seja de alto-nível.

No capítulo nove desta tese, estes desafios serão analisados com maior detalhe e novas metodologias de coordenação adequadas ao domínio do futebol robótico simulado serão propostas.