PAULO HENRIQUE GAMA E SILVA

AJUSTES NO TIME UFLA2D NO DOMÍNIO DE FUTEBOL DE ROBÔS

LAVRAS – MG

2012

PAULO HENRIQUE GAMA E SILVA

AJUSTES NO TIME UFLA2D NO DOMÍNIO DE FUTEBOL DE ROBÔS

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador

Prof.Dr. Joaquim Quinteiro Uchôa

LAVRAS – MG

2012

PAULO HENRIQUE GAMA E SILVA

AJUSTES NO TIME UFLA2D NO DOMÍNIO DE FUTEBOL DE ROBÔS

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 31 de Outubro de 2012.

Profa.Dra. Ana Paula Piovesan Melchiori	UFLA
Prof.Dr. Wilian Soares Lacerda	UFLA

Prof.Dr. Joaquim Quinteiro Uchôa
(Orientador)

LAVRAS – MG

2012

Dedico este trabalho à Deus, à minha família e aos amigos.

RESUMO

O futebol de robôs surgiu como uma alternativa educacional e de pesquisa para o desenvolvimento de Inteligências Artificiais e Robótica Inteligente. Apesar de aparentemente ser um simples "brinquedo", a competição entre robôs visa o aperfeiçoamento do controle individual de robôs móveis e da coordenação entre múltiplos robôs. Este trabalho descreve a simulação em duas dimensões do futebol de robôs, detalhando a estrutura do ambiente de simulação, do ambiente de desenvolvimento e do futebol de robôs, objetivando o incentivo aos estudantes e pesquisadores da área. Para isto, realizou-se diversos testes e análises para subsidiar equipes que desejam ingressar nesse ramo, sendo apontadas características positivas e negativas dos times-base mais utilizados. Ao final da bateria de testes, concluiu-se que o melhor time básico a ser adotado é o *HELIOS_base* que obteve os melhores resultados nos confrontos diretos nas partidas.

Palavras-Chave: *RoboCup Soccer Simulation 2D*, Futebol de Robôs, UFLA2D, Inteligência Artificial

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O futebol é um dos jogos mais praticados no mundo e possui como principal característica a cooperação entre os jogadores de um mesmo time para alcançar a vitória. Por ser um ambiente imprevisível, não determinístico e dinâmico, tem sido considerado um problema padrão para a Inteligência Artificial e gerado vários tópicos de pesquisa em diversas áreas, tais como: sistemas com múltiplos agentes, algoritmos de cooperação, inteligência distribuída, aprendizagem, reconhecimento de padrões e robótica inteligente (SILVA; SIMÕES; ARAGÃO, 2007).

Os avanços das técnicas observados no futebol de robôs têm um grande potencial para outras áreas como, por exemplo: *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 autônomos (*drone*), entre outros (JR; BIANCHI; MATSUURA, 2011). A liga simulada de futebol de robôs é um domínio que engloba muitos desafios como, por exemplo, a simulação em tempo real, informação sensorial (auditiva, visão e física). Ela é baseada em um modelo energético realista, com agentes heterogêneos, ações cooperativas complexas, transformação de ações de baixo nível em comandos de alto nível, ambiente multiobjetivo, parcialmente adverso e parcialmente cooperativo, entre outros.

1.1 Contextualização

A crescente demanda por sistemas mais robustos e eficientes, bem como as atuais exigências e restrições aos tempos de resposta, tem fomentado a aplicação e a incorporação de técnicas de Inteligência Artificial, pois estas são capazes de tratar problemas cuja complexidade não permite a utilização de técnicas tradicionais. Visando incentivar as pesquisas nesta área e em áreas correlatas, tais como siste-

mas de tempo real e visão computacional, o futebol de robôs foi proposto como um problema padrão, onde técnicas e estratégias podem ser testadas e comparadas e posteriormente transferidas para outras áreas (MACKWORTH, 1993).

O futebol de robôs apresenta-se como um ambiente dinâmico e complexo, propício ao estudo e desenvolvimento de técnicas avançadas, assim como despertar a curiosidade e atrair as pessoas, que de forma lúdica, estudam os mais avançados meios e tecnologias para desenvolverem trabalho em equipe.

Alan Mackworth, professor da *British Columbia University* (Canadá) teve a ideia de realizar um torneio de futebol entre robôs no ano de 1992 (CHEN *et al.*, 2003). No mesmo momento, era discutido os grandes desafios para a Inteligência Artificial em um *Workshop* no Japão, tentando idealizar o futebol de robôs para a aplicação das técnicas inteligentes existentes na época, assim como Alan Mackworth.

No ano de 1993, foi criada a Robot *J-League* em alusão ao campeonato japonês (*J-League*) de futebol. Nesta primeira competição foi sustentado o surgimento da primeira confederação de futebol robótico, a *RoboCup* (*Robot World Cup Soccer Games and Conferences*) (ROBOCUP, 2011). A primeira competição oficial organizada pela *Robocup* com jogos e conferências foi realizada no ano de 1997 com grande sucesso, mais de 40 equipes participaram, contando com a presença de 5000 expectadores.

1.2 UFLA2D

As atividades do time UFLA2D, uma equipe do futebol de robôs simulado, iniciaram-se no Laboratório de Inteligência Computacional e Sistemas Avançados (**LICESA**), situado no Departamento de Ciência da Computação na Universidade Federal de

Lavras (**UFLA**). O UFLA2D surgiu como um projeto de iniciação científica de alunos do curso de Ciência da Computação.

Este time foi criado no ano de 2010 pela aprovação e suporte financeiro da FAPEMIG[1] através de um projeto[2] apoiado pelo governo estadual a fim de promover a pesquisa acadêmica e tecnológica, na execução de projetos de cunho prático e que possibilitariam a participação das equipes discentes em competições tecnológicas de caráter educacional. Inicialmente era formado por uma equipe com 4 integrantes, sendo três deles alunos de graduação e uma coordenadora/professora. Hoje conta com um aluno de graduação, autor deste trabalho, um orientador e uma colaboradora.

1.3 Objetivos

A *RoboCup* visa encorajar e subsidiar pesquisas na área de inteligência computacional e robótica inteligente, com o objetivo de capacitar robôs para a realização autônoma de atividades e tarefas sendo elas singulares ou cooperativas. Assim, este trabalho visa estudar, entender e apontar melhorias para o desenvolvimento de um sistema multi agente forte, rápido e equilibrado.

Este trabalho fomenta trabalhar com conhecimentos adquiridos por meio da observação de partidas, estudo do código e aplicação de técnicas do futebol real a fim de alcançar um bom toque de bola, precisão nos chutes, forte marcação, escolha correta das melhores jogadas. Assim, sendo destacadas as melhorias necessárias que possam embasar a aplicação de técnicas de inteligência computacional e

[1] Fundação de Amparo à Pesquisa do Estado de Minas Gerais

[2] Projeto Santos Dumont - Objetiva financiar projetos de iniciação tecnológica que permitam ao aluno testar seus conhecimentos e possibilitar a participação das equipes em competições tecnológicas de caráter educacional.

otimização, com o intuito na melhora da tomada de decisão com ou sem a bola, individual ou coletivamente.

Também é vislumbrado neste estudo contribuir para a ampliação do grupo de pesquisa na Universidade Federal de Lavras (UFLA) para outros departamentos da área de inteligência computacional. Possibilitando a aquisição de conceitos para outras categorias da *RoboCup* que são disputadas com robôs reais.

1.4 Metodologia

Para o desenvolvimento deste trabalho fez-se necessário a utilização do *framework agent2D* versão **3.1.1**, disponível gratuitamente em (TEAM, 2012). Ele foi escolhido por ter tido excelentes resultados nas últimas edições do campeonato mundial *RoboCup Soccer Simulation 2D* e em testes realizados neste trabalho. Esse *framework* é extremamente modularizado e com resultados impressionantes sobre os outros times-base testados e com a credibilidade adquirida de conquistas importantes somada ao fato de ser constantemente atualizado.

Pretende-se também utilizar o material disponibilizado pelos desenvolvedores do time *Helios* como base de pesquisa bibliográfica. Para realizar os testes medindo a eficiência das alterações serão realizadas várias partidas contra times que disputaram a última edição do campeonato mundial *RoboCup Soccer Simulation 2D*.

1.5 Estrutura do Texto

Este texto foi estruturado visando o melhor entendimento dos leitores. Com isso, a primeira parte do texto (Capítulo 2) apresenta as principais categorias do futebol de robôs, detalhando-as. Após mostrar as modalidades existentes, aprofunda-se

no tema deste trabalho que é a liga simulada em duas dimensões. O texto mostra o detalhamento envolvendo o servidor de jogo e suas estruturas internas (servidor e monitor de jogo), contendo as regras de jogo, comunicação, comportamento e treinador.

No Capítulo 3, foi feita uma análise sistemática dos times-base disponíveis, sendo apontado os pontos fortes e pontos fracos de cada equipe, e ao final um demonstrativo das partidas disputadas entre eles.

Para finalizar este trabalho são apresentadas seções descrevendo as alterações feitas, análise de desempenho das alterações, resultado das partidas e comparativo com os times sem alterações (times-base).

2 FUTEBOL DE ROBÔS

A principal competição da RoboCup é o futebol, onde fomenta-se as pesquisas em ambientes cooperativos multi-robôs e sistemas multi-agentes nas adversidades de um ambiente dinâmico, no qual todos os robôs são totalmente autônomos, conforme descrito em (FRACCAROLI, 2011).

Dentro da *RoboCup*, existem diversas categorias para o futebol de robôs, as quais serão detalhadas nas subseções a seguir.

2.1 *Robocup Humanoid*

A categoria *humanoid*, Figura 2.1, possui até dois robôs bípedes para cada time, e visa a pesquisa em áreas como: visão computacional, auto localização, comunicação, entre outras. Esses robôs tem o corpo semelhante ao corpo humano tanto nas características físicas como sensoriais.

Figura 2.1: Categoria *Robocup Humanoid* – Fonte: (ROBOCUPGALLERY, 2011)

2.2 *Robocup Middle Size*

Na categoria *middle size*, Figura 2.2, cada time pode ter 4 robôs com até 75 cm de altura e 50 cm de diâmetro, onde todos os sensores são do tipo *on-board* e visa a pesquisa em áreas como: auto localização, visão computacional, integração de sensores, percepção distribuída e controle de movimentos do robô.

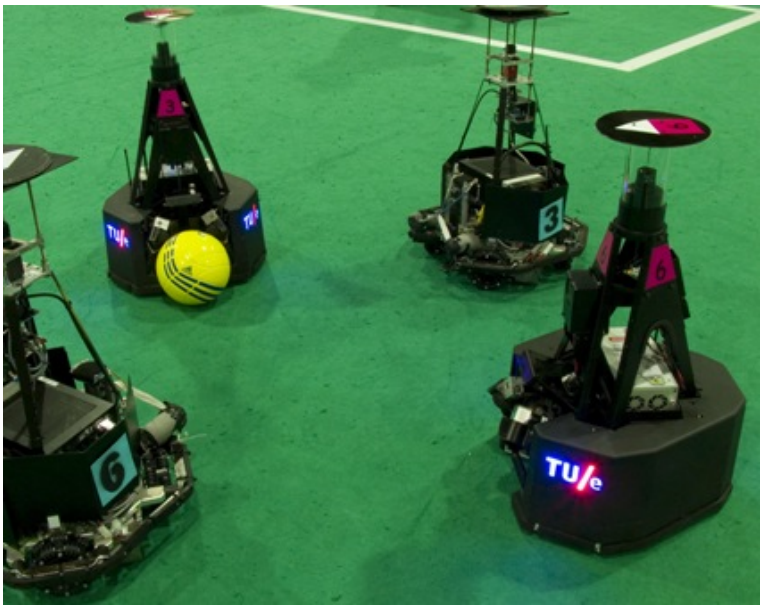

Figura 2.2: Categoria *Robocup Middle Size* – Fonte: (DESIGN, 2011)

2.3 *Robocup Small Size*

A categoria *small size (F180)*, Figura 2.3, contém times com até 5 jogadores e que pode ter até 15 cm de altura e 18 cm de diâmetro. E as áreas de pesquisa existentes são as mesmas presentes na categoria *middle size*, diferenciando apenas no tamanho dos robôs. O foco é dado no problema de inteligência cooperativa multi-robôs/agentes e controle em um ambiente altamente dinâmico.

Figura 2.3: Categoria *Robocup Small Size* – Fonte: (ROBOCUPGALLERY, 2011)

2.4 *Robocup Standard Platform*

A categoria *standard platform*, Figura 2.4, contém robôs idênticos em cada equipe diferindo apenas no *software* desenvolvido por cada equipe, sendo os robôs totalmente autônomos e não há controladores externos, nem por interferência humana nem por interferência computacional. A plataforma usada atualmente são os humanoides *NAO*, desenvolvidos pela empresa francesa **Aldebran Robotics** (ALDEBRAN, 2011).

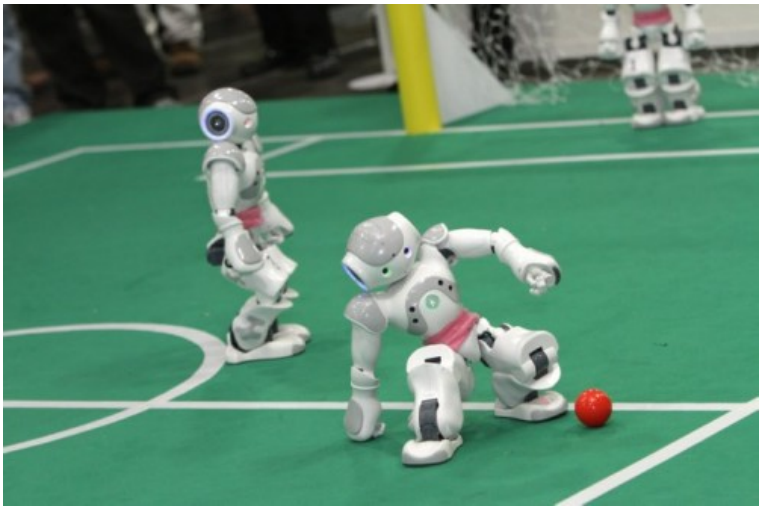

Figura 2.4: Categoria *Robocup Standard Platform* – Fonte: (ROBOCUPGALLERY, 2011)

2.5 *Robocup Simulation*

A categoria *simulation*, Figuras 2.5 e 2.6, é dividida em duas subcategorias: duas e três dimensões. Na simulação enfoca-se na parte de implementação das técnicas de inteligência computacional para a autonomia dos agentes com trabalho em equipe e análise de situações de jogo para tomar a decisão correta.

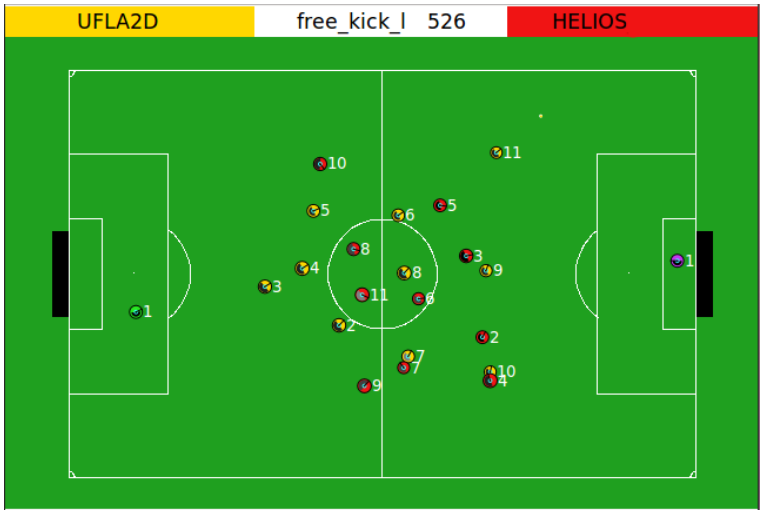

Figura 2.5: Categoria *Robocup Simulation 2D*

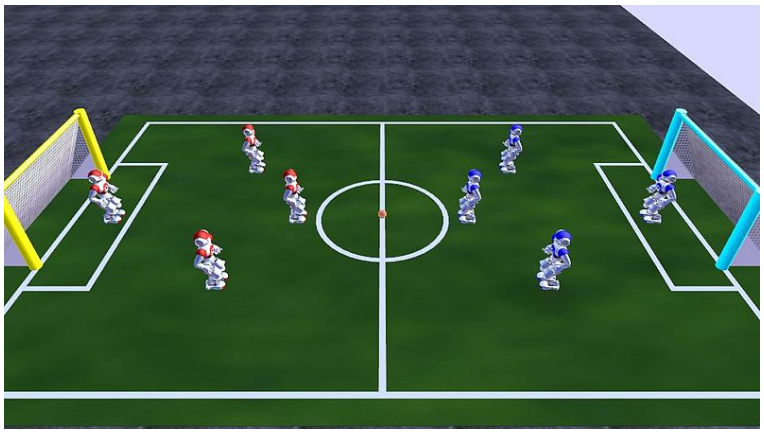

Figura 2.6: Categoria *Robocup Simulation 3D* – Fonte: (WIKIPEDIA, 2012)

A liga simulada em três dimensões possibilita a competição entre robôs humanoides em uma simulação realista tanto nas regras quanto na parte física de uma partida de futebol. A plataforma se esforça para reproduzir os desafios de programação de *software* enfrentados ao construir robôs físicos reais para o futebol (SIMULATION, 2012).

A simulação em 3D é executada no *software RoboCup Simulated Soccer Server 3D (rcssserver3d)* que é multiplataforma. Ele roda em *Linux, Windows e Mac OS X*. O mecanismo de simulação subjacente é o *SimSpark*[1]. Os agentes são controlados por processos externos e a comunicação com o servidor é feita via TCP, na porta 3100 por padrão. Já o servidor envia informações do estado do jogo para cada um dos agentes, e em resposta, o agente envia comandos para a simulação que controla o movimento do corpo do agente. O *rcssserver3d* não tem uma interface gráfica própria, por isso, existe um monitor dedicado que conecta com o servidor via TCP e recebe informações sobre o estado da partida. O monitor padrão é denominado *rcssmonitor3d* e é capaz de reproduzir gravações de partidas através de arquivos de log.

A simulação em duas dimensões é disputada por dois times com onze agentes de linha e um agente técnico. Cada agente representa um jogador de futebol que recebe vários tipos de informações limitadas contendo diversas situações de jogo, com alguns ruídos propositalmente introduzidos pelo servidor do simulador do jogo. Com isso, cada jogador deve ser capaz de inferir sob a melhor estratégia no momento, seja ela individual ou coletiva, assim como no futebol convencional. Disputada no Brasil desde 2005, as regras de jogo são muito semelhantes às de uma partida real, mantidas pela *Féderation Internationale de Football Association* **(FIFA)**. Também são utilizadas algumas regras específicas da *RoboCup* como em um caso que haja um time dopando o servidor com mensagens a fim de atrasar o jogo.

A liga simulada *2D* está entre as categorias mais antigas da *RoboCup*, sendo que o foco é dado na inteligência artificial e estratégias para o time. Pode-se dizer

[1] *SimSpark* é um sistema genérico de simulação multiagente. Ele suporta o desenvolvimento de simulações físicas para Inteligência Artificial e pesquisa em robótica como um *framework* de aplicação *open-source*. É comumente utilizada na pesquisa acadêmica e na educação (SIMSPARK, 2012).

que a simulação é a porta de entrada de equipes no futebol de robôs, por ter um custo bem mais baixo e ser menos complexa em relação às outras.

A partida disputada no futebol de robôs simulado em duas dimensões tem duração de 6000 ciclos, que são aproximadamente 5 minutos, divididos em dois tempos de 3000 ciclos. Após o início da partida os times devem ser capazes de tomar suas próprias decisões e adotar suas estratégias previamente programadas, não sendo permitida qualquer interferência humana após o início do jogo. Somente o agente técnico pode alterar o time, como alterar a posição dos atletas, substituir atletas cansados, entre outras opções. Porém, o técnico tem a restrição de realizar ações, exceto comunicação, quando a partida está parada (intervalo na cobrança de faltas, lateral, tiro de meta, impedimentos, gols e intervalo da partida).

Para o desenvolvimento de um time existem duas opções, sendo possível trabalhar em cima de uma implementação básica (chamada de time-base) ou desenvolvendo todo o código do time. Uma das vantagens de se utilizar o time-base é o aproveitamento das implementações de baixo-nível já desenvolvidas, que são criadas por equipes mais experientes.

No time desta modalidade têm-se uma vasta opção de variáveis que podem ser exploradas como: velocidade do jogador e da bola, posicionamento em campo, distâncias relativas e globais, precisão, força, energia dos atletas. Com isso, esse ambiente de simulação em duas dimensões torna-se um grande meio de pesquisa com diversas aplicações em um mundo real.

2.6 *Robocup Soccer Simulator*

O simulador *RCSS* (ROBOCUPPROJECT, 2012) é um sistema escrito para permitir o suporte ao jogo de futebol, onde competem múltiplos agentes virtuais em um ambiente incerto e em tempo real. Uma de suas vantagens é a forma de abstração,

que permite os desenvolvedores um manuseio dos problemas robóticos como reconhecimento de objeto, comunicação, auto-localização, entre outros. A abstração permite aos pesquisadores focarem nos conceitos de alto-nível como cooperação e aprendizagem.

O *RCSS* é dividido em duas partes, sendo que a comunicação com os clientes, controle das regras de jogo e simulação dos movimentos é de responsabilidade do *SoccerServer*. A parte de visualização da partida, visualização do campo, dos jogadores e bola é de responsabilidade do *SoccerMonitor* (CHEN *et al.*, 2003). A seguir é mostrada na Figura 2.7 a arquitetura do *RCSS*, sendo detalhado nas seções subsequentes seus principais elementos.

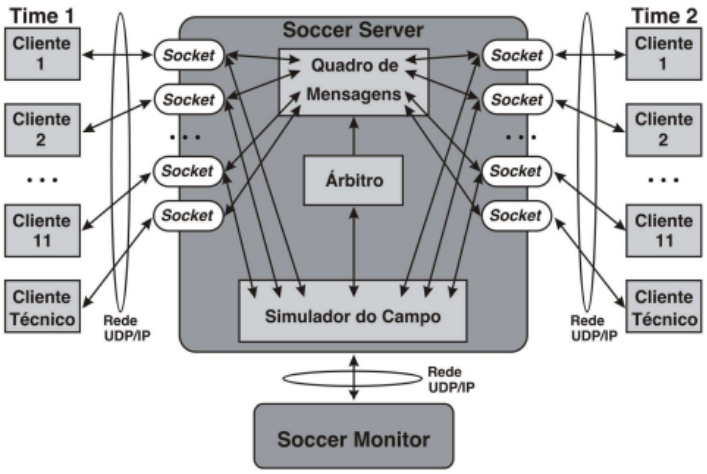

Figura 2.7: Arquitetura do simulador *2D* (REIS, 2003)

2.6.1 *SoccerServer*

Uma partida é realizada ao estilo cliente/servidor, cada cliente controla os movimentos de um jogador. A comunicação entre eles é realizada através de soquetes *UDP/IP*. Depois de estabelecida a conexão, o cliente envia sinais de controle para um jogador e recebe de volta informações dos sensores áudio-visuais do joga-

dor. Estes sinais sofrem propositadamente a intervenção do servidor que acopla às informações ruídos, para dificultar ainda mais o ambiente, testando mais profundamente o agente.

2.6.2 *SoccerMonitor*

O monitor permite visualizar o campo de jogo, sendo possível conectar vários monitores em um único servidor[2]. Isso é muito útil para analisar a partida de forma mais real. Várias opções podem ser configuradas a fim de informar os telespectadores sobre o cansaço dos jogadores, linha de impedimento, visão dos jogadores, entre outras.

Ele conta também com uma ferramenta muito importante que é o *logplayer*, que realiza a gravação das partidas realizadas, permitindo a visualização com opções de pausa, adiantar, voltar, câmera lenta, assim o desenvolvedor tem a possibilidade de acompanhar os jogos detalhadamente.

2.6.3 Regras

As regras do futebol simulado são muito semelhantes às regras do futebol do mundo real; um conhecimento básico sobre as regras do futebol real é, portanto, fundamental. Para que os jogadores se orientem num campo de futebol simulado existem marcações em torno e dentro do respectivo campo. A Figura 2.8 mostra o campo virtual com as dimensões proporcionais às do futebol real (105 m x 68 m) com as respectivas marcações.

Todos os marcos e linhas que se vêem na Figura 2.8 servem para auxiliar os jogadores dentro do campo. Por exemplo **(*flag* c)** diz respeito ao marco que se encontra no centro do campo de futebol virtual; **(gol l)** e **(gol r)** referem-se ao centro

[2] *Software* que realiza o jogo, comunica com os times, mantém a interface gráfica, etc.

da baliza do lado esquerdo e ao centro da baliza do lado direito respectivamente; (*line* t), (*line* b), (*line* r) e (*line* l) dizem respeito às delimitações do campo de futebol simulado (105m x 68m). Cada marco é identificado por uma etiqueta durante a comunicação simulador-jogador.

Figura 2.8: *Flags* de orientação

O árbitro virtual, contido no servidor, faz cumprir as regras especificadas para este domínio e comunica com os jogadores através de mensagens. Para supervisionar o jogo, um árbitro humano também está presente para casos extraordinários.

2.6.4 Comunicação

A comunicação entre clientes e servidor é regida por um conjunto de protocolos. Os protocolos são conexão, ação e percepção. O protocolo de conexão permite aos clientes ligarem-se e desligarem-se do servidor conforme descrito na Figura 2.9.

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**)	

Figura 2.9: Protocolo de Conexão – Fonte: (REIS, 2003)

No protocolo de ação é definido a sintaxe das mensagens de comando que os clientes podem enviar para o servidor, assim como as respectivas respostas, conforme demonstrado na Figura 2.10.

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 <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)

Figura 2.10: Protocolo de Ação – Fonte: (REIS, 2003)

Os jogadores percepcionam o ambiente a três níveis: auditivo (*hear*), visual (*see*) e físico (*sense_body*). A Figura 2.10 mostra ainda as mensagens de comando,

de informação sensorial que o servidor pode enviar aos clientes. Após esta recepção de mensagens os clientes, de acordo com os seus campos de visão e de audição, conseguem ter noção daquilo que se passa no terreno e executam as ações necessárias para cumprir o seu papel.

2.6.5 Comportamento das Ações dos Agentes

Para os agentes descobrirem quando devem efetuar um chute ao gol, tocar a bola ao companheiro ou tomar a bola do adversário, primeiro devem conhecer o ambiente à sua volta e quais as possibilidades de movimentos existentes. Conforme descrito no protocolo de ação (Figura 2.10), há três tipos de sensores que permitem aos jogadores realizarem essas ações com eficiência conforme cada tipo de jogador.

O sensor visual (*see*) detecta as distâncias e as direções dos objetos e jogadores. O sensor auditivo (*hear*) detecta as mensagens enviadas pelo árbitro, treinador, colegas de time e adversários. A audição dos jogadores, nas competições, está limitada a uma distância máxima de 50 metros, exceto para os treinadores e árbitro que não têm limites de distância para serem ouvidos. O sensor físico (*sense_body*) detecta o estado do jogador, incluindo a sua energia, velocidade e ângulo do pescoço relativamente ao corpo.

Para os jogadores executarem ações, o simulador tem comandos parametrizáveis à disposição dos jogadores durante o período de jogo de futebol simulado. Existem quatro tipos de ações principais: movimento (*dash*, *turn* e *move*); interação com a bola (*kick*, *tackle* e *catch*); controle de percepção (*turn_neck*, *attentionto*, *change_view*) e comunicação (*say*, *pointto*). Os agentes com a bola, por ciclo, só podem enviar um comando do tipo movimento ou interação por ciclo. Esta restrição tem todo o sentido, pois, por exemplo, um jogador quando chuta não pode estar se movimentando ao mesmo tempo. Caso sejam enviados comandos que violem as restrições, o servidor escolhe um aleatoriamente.

O servidor aceita comandos de baixo-nível dos jogadores (*turn, dash, kick, catch,turn_neck*, entre outros), executa-os de forma imperfeita e envia informação sensorial (com erros) aos jogadores. Os modelos de ruído dos sensores e atuadores são inspirados em sistemas robóticos típicos. No entanto, outras características, tais como as regras do jogo, energia limitada, comunicação limitada, movimento e visão, são inspiradas nas limitações humanas. O treinador dispõe da informação completa e livre de erros do jogo, sendo impedido de agir sobre o mesmo, pois, ele pode unicamente comunicar-se com os jogadores quando o jogo se encontra parado (antes do início do jogo, na marcação de escanteios e faltas, entre outros momentos de paralisação).

Devido ao fato da informação sensorial disponível aos jogadores ser muito limitada para tentar imitar os jogadores do futebol real (cone de visão de 45 graus e comunicação limitada a 10 *bytes* por ciclo de simulação), para serem bem sucedidos nas suas ações é conveniente que os agentes de uma equipe, efetuem previsões de quais as ações que os adversários (e colegas de equipe) irão realizar. Desta forma é possível, por exemplo, chutar ao gol, de costas para este, sem ver o goleiro, prevendo onde ele estará numa determinada posição, ou passar a bola a um colega de equipe, sem o ver, prevendo qual será a sua posição no campo na situação atual.

2.6.6 Treinador

O treinador tem como função guiar a equipe de futebol de forma a conseguir a vitória. É responsável por escolher a melhor tática e por efetuar as substituições necessárias ao melhor rendimento da sua equipe. Em uma partida o treinador recebe mensagens dos jogadores, do árbitro e pode enviar mensagens auditivas, sem restrição de distância máxima, a todos os jogadores da sua equipe.

2.7 Comentários Finais

Este capítulo apresentou as categorias futebolísticas existentes na *RoboCup*, enfocando-se na simulação em duas dimensões, que é o alvo principal deste trabalho. No capítulo subsequente, será visto os times-base disponibilizados sem custos por equipes com grande experiência nesta área e que desejam subsidiar equipes novas com seus conhecimentos.

3 TIMES-BASE

O time-base é uma plataforma arquitetada para abstrair códigos de baixo-nível e auxiliar outras equipes, mas é opcional o seu uso, visto que cada equipe pode desenvolver sua própria estrutura. Este capítulo objetiva apresentar alguns dos times-base mais utilizados no cenário mundial apontando suas principais características.

3.1 *UvA TriLearn*

UvA TriLearn foi tema de uma dissertação de mestrado (BOER; KOK, 2002) que baseou-se na divisão arquitetural em três camadas, tomando que tarefas complexas podem sempre decompor-se hierarquicamente em várias sub-tarefas simples. Esta arquitetura mostrada na Figura 3.1 é hierárquica no sentido em que ela contém três camadas em diferentes níveis de abstração.

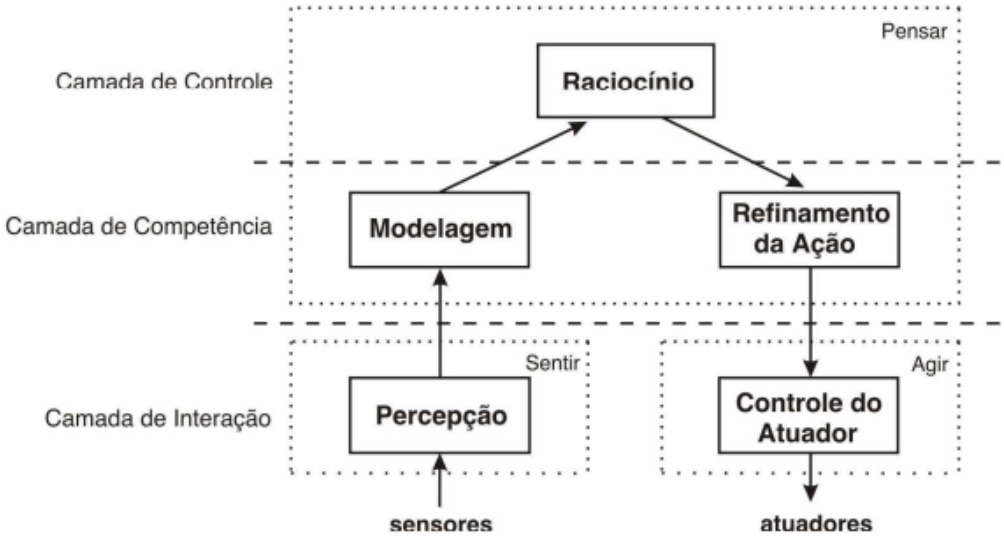

Figura 3.1: Arquitetura Hierárquica em três camadas – Fonte: (BOER; KOK, 2002)

A camada mais baixa (Camada de Interação) é a parte responsável pelas interações com o ambiente de simulação do servidor, com a entrada no sistema através de sensores e a saída (ação) nos atuadores (agentes). Ela é capaz de abstrair o máximo de detalhes do servidor possível para as outras camadas superiores. A camada intermediária (Camada de Competência) usa a funcionalidade oferecida pela camada de interação a fim de construir um modelo abstrato do mundo e implementar várias habilidades para cada agente. A camada do topo (Camada de Controle) contém os componentes de raciocínio do sistema, na qual é feita a escolha da melhor estratégia para o momento corrente.

O time-base *UvA TriLearn* foi fornecido por estudantes da *Universiteit Van Amsterdam* [1] que após a conquista da *Robocup* 2003, disponibilizou parte de seus códigos, retirando apenas a camada de controle (como visto na Figura 3.1 acima)

[1] Universidade de Amsterdã

para que interessados na competição não tivessem a necessidade de implementações de baixo-nível como conexões e troca de mensagens com o servidor.

Este time-base contêm três formações táticas disponíveis: (3-4-3) que é mais defensiva, (4-4-2) a mais equilibrada e (4-3-3) que é mais ofensiva, sendo que por padrão é adota a terceira tática. Na formação **4-3-3**, por exemplo, têm-se quatro jogadores com função de zagueiros, três jogadores atuando no meio-campo e três atacantes bem espalhados, o que torna o poder ofensivo bem forte.

A única ação dos agentes, quando estão com a posse de bola, é realizar o cálculo aproximado da posição do gol adversário e chutá-la com a maior força possível nessa direção. Não há ação bem elaborada e/ou coletiva, pois, nessa arquitetura tais ações são criadas na camada de controle (Figura 3.1), inexistente na versão *opensource* deste time.

3.2 *WrightEagle Base*

O time *WrightEagle* (2D, 2012) da China, que disputa a *RoboCup Simulation* desde o ano 1999 e nos últimos 3 anos ganhou duas vezes a *RoboCup*, disponibiliza parte de seus códigos para a comunidade contribuindo para que possam preocupar menos com a implementação de baixo-nível e focar na parte estratégica e inteligente para a disputa de uma partida.

Este *framework* possui um comportamento básico munido com redes neurais artificiais para a tomada de decisão e também com muitas implementações baixo-nível. É bem modularizado sendo seu código separado por comportamento (*Behavior*) e por ações (*Action*), mas possui pouca documentação.

3.3 *Helios_base*

O time *Helios* do Japão que disputa a *RoboCup Simulation* desde o ano 2000 e que nos últimos 3 anos chegou a final da *RoboCup*, disponibiliza parte de seus códigos (chamado de plataforma de desenvolvimento *agent2D*) sob licença *GNU GPL* com a intenção de contribuir com outras equipes e com a sociedade acadêmica. Os integrantes do time são responsáveis também pelo *RoboCup Soccer Simulator (RCSS)*, o que facilita a comunicação do time com o servidor.

O *framework agent2d* já tem um comportamento básico para os agentes. Roda sob a biblioteca *librcsc*[2] e seu código é separado por comportamento (*Behavior*) e ações (*Action*).

A modelagem deste time é feita seguindo a estrutura conhecida por *Action Chain*, que significa cadeia de ações. Esta cadeia funciona de forma similar a um grafo, onde cada nó é constituído por um par Ação/Estado. A ação é algo em que o agente pode realizar como um passe, um chute, um drible. Já o estado é a previsão do modelo do mundo após a ação ser consumada.

A modelagem *Action Chain* objetiva uma sequência de previsões que possam consolidar a obtenção de um resultado alvo, que se dá por meio de ações individualistas e/ou cooperativas. Por exemplo, se o agente prevê que uma sequência de 3 jogadas irá levar ao gol, ele começa a executar estas jogadas.

A dificuldade de consolidar tais ações está na aleatoriedade e dificuldade que o ambiente do futebol proporciona. Em um jogo de xadrez, por exemplo, as ações são bem definidas e o único fator de caos são as decisões do oponente. Já no futebol, diversos fatores podem fazer as previsões estarem erradas, e por isso as previsões do *Action Chain* tem que admitir certas falhas. Acoplar um estado a

[2]Uma biblioteca base para desenvolver um programa cliente para *RCSS* e que pode ser usada como uma estrutura para um time de futebol simulado.

cada ação da cadeia é o que faz o processo se tornar inteligente. Sem isso, seria simplesmente a descrição dos movimentos a serem seguidos pelo agente.

A parte da estrutura do time que consolida as ações é nomeada *Behavior*, que significa comportamento. Nesta parte do time as decisões são subdivididas a fim de modularizar o código, onde decisões como passar a bola diretamente ao companheiro podem ser divididas minunciosamente em partes menores como, escolher um alvo, apontar para ele, mostrar a intenção de tocar e verificar se seu companheiro recebeu a bola.

A estrutura *Behavior* é responsável também por ações mais bem elaboradas, como realizar tabela entre jogadores, passes em profundidade, entre outras.

3.4 UFLA2D

Foi adotado o time-base *UvA TriLearn* (BOER; KOK, 2002), pelo fato do mesmo ter um código bem definido, documentado e que servia como base para a maioria dos times brasileiros na época. Inicialmente foram realizadas alterações no time como implementação do passe, diminuição da área de atuação do goleiro, chutes mais bem elaborados, entre outras.

O passe foi implementado seguindo o raciocínio mais usado no meio futebolístico, tocar para o companheiro com melhores condições de jogo. Assim, fazia-se uma análise de todos os aliados em campo exceto o goleiro e passava a bola para aquele em que seu estado fosse o melhor dentre os demais.

A necessidade da diminuição da área de atuação do goleiro fez-se necessária após a análise de várias partidas realizadas contra times mais bem elaborados, em que a facilidade dos gols adversários marcados foi grande. Com isso, foi feita a alteração no tamanho do retângulo de movimentação padrão do agente goleiro

quando estava na eminência de uma situação de risco, tendo uma melhor eficácia nas tentativas de interceptação de jogadas adversárias pelo goleiro.

Em relação ao chute dos jogadores, foi elaborada uma estratégia que tentava analisar a posição do goleiro adversário no momento do arremate ao gol. Com o cálculo da posição do goleiro consolidado, o agente com boa posição em relação ao gol adversário tentava efetuar o chute na posição contrária ao goleiro oponente, melhorando assim a eficiência dos arremates.

Com essas alterações realizadas, surgiu então a necessidade de estratégias e cálculos mais bem elaborados, como o cálculo das condições dos atletas para receberem o passe, cálculo da posição boa para o chute e melhoria na marcação. Após a realização de uma pesquisa teórica para decidir qual técnica inteligente utilizar, foi decidido por utilizar um sistema híbrido, composto de um Sistema *Fuzzy* e Redes Neurais Artificiais, que estava entre as técnicas mais difundidas entre o mundo da simulação de futebol (ALCANTARA; SILVA, 2011).

A Figura 3.2 exemplifica a modelagem do cálculo do grau de pertinência para a distância do gol adversário no sistema *Fuzzy*.

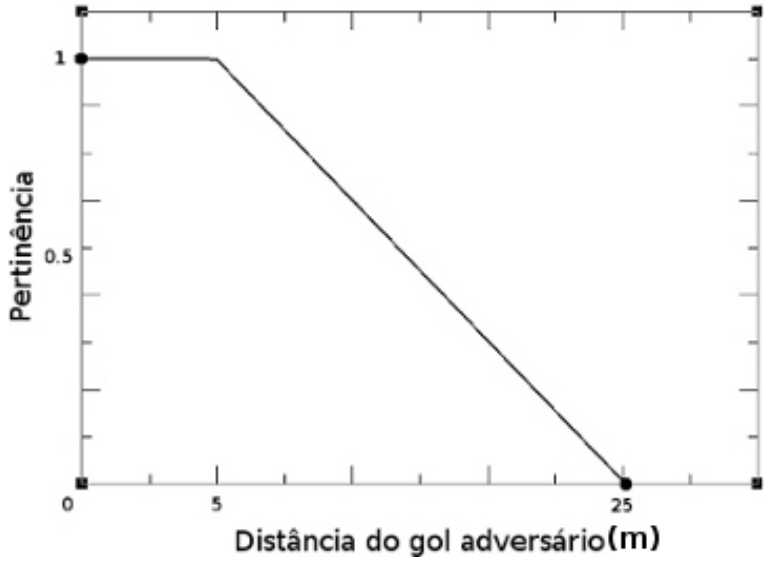

Figura 3.2: Modelagem *fuzzy* usada para representar a distância do gol adversário

Após a obtenção dos valores representantes das situações de jogo, passava-se como entrada da Rede Neural Artificial previamente treinada, que em contrapartida retornava a ação correta para aquele momento, conforme representado na Figura 3.3.

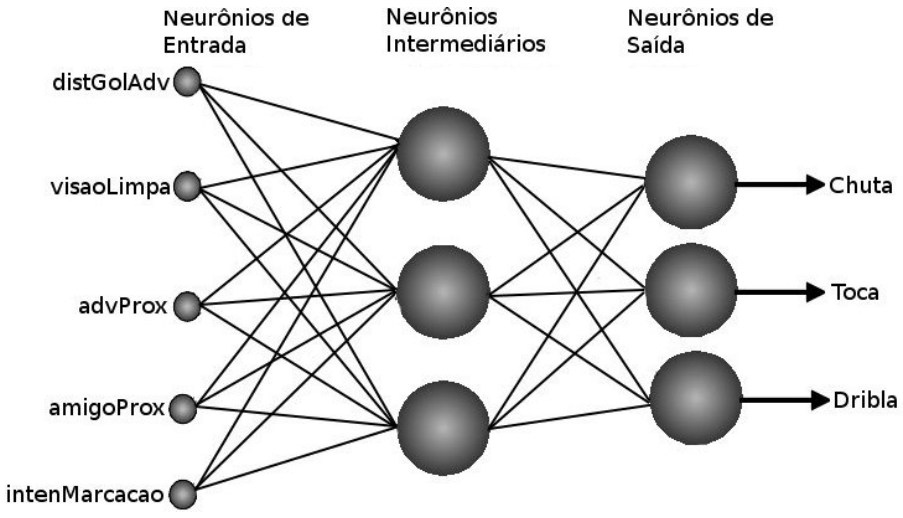

Figura 3.3: Modelagem do Sistema Híbrido utilizado

A representação real de possíveis situações de jogo em que o sistema híbrido seria acionado é mostrado nas Figuras 3.4 e 3.5.

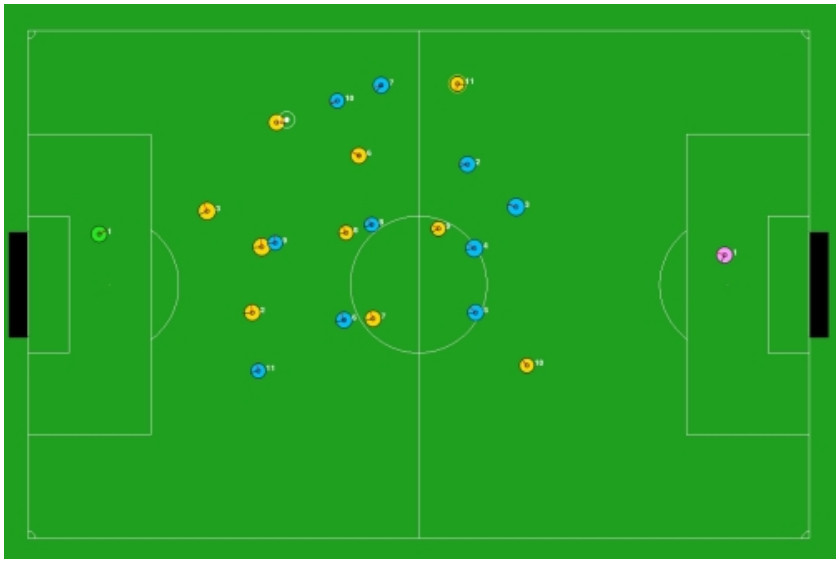

Figura 3.4: Situação para tomada de decisão com agente zagueiro

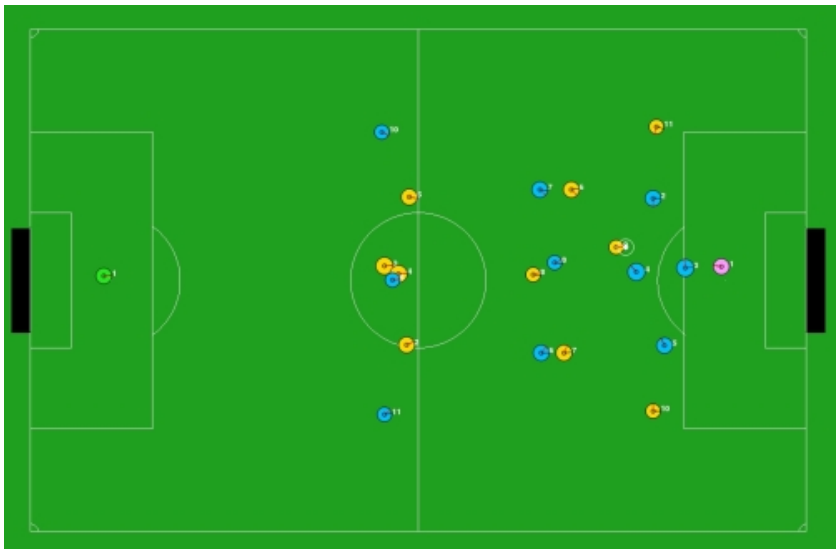

Figura 3.5: Situação para tomada de decisão com agente atacante

A implementação de parte deste sistema permitiu refinar as alterações realizadas anteriormente, com cálculos de aproximações nas posições, análise geral da situação de jogo e separação das ações realizadas por categorias dos atletas como goleiro, jogadores da zaga, jogadores de meio-campo e jogadores de ataque conforme descrito abaixo. Foi utilizada apenas a parte *Fuzzy* desta modelagem na estrutura do time, pois, surgiram novas possibilidades de implementações mais eficientes descritas neste trabalho.

```
/* UFLA 2D goalie player */
SoccerCommand Player::ufla2d_goalie(  )
{
   ...
}
/* UFLA 2D defender players */
SoccerCommand Player::ufla2d_defender(  )
{
   ...
}
/* UFLA 2D Midfielders players */
SoccerCommand Player::ufla2d_midfielder(  )
{
   ...
}
/* UFLA 2D attackers players */
SoccerCommand Player::ufla2d_attacker(  )
{
   ...
}
```

Figura 3.6: Modularização das ações

Essa modularização nas categorias dos atletas permitiu a criação de ações específicas para cada tipo de jogador, conforme descrito na Figura 3.6.

O time participou de poucas competições/eventos, sendo apenas uma abrangendo as equipes nacionais. Na 1ª Copa Mineira de Simulação 2D realizada na cidade de São João Del Rei no ano de 2011, o time conseguiu o segundo lugar, realizando boas partidas. Neste mesmo ano foi realizada também na cidade de São João Del Rei, a Competição Brasileira de Robótica (CBR) com a participação de oito equipes e algumas com experiência mundial. O time UFLA2D obteve o quinto lugar na classificação geral, sendo o mais bem colocado dentre aqueles que não utilizavam o *Agent2D* – time-base desenvolvido pela equipe *Helios* (AKIYAMA *et al.*, 2011)– como *framework* de desenvolvimento.

3.5 Comparativo entre os times-base

A fim de comparar os times-base para auxiliar na escolha do que deveria ser adotado no UFLA2D, foram realizadas um total de quinze partidas entre as equipes: *UvA TriLearn 2003*, *Helios-base* e *WrightEagle-BASE* sem nenhuma alteração nas equipes, ou seja, do modo em que são disponibilizadas pelos seus desenvolvedores. Cada equipe realizou cinco partidas contra cada uma rodando no ambiente *rcssserver 15.0.0* em uma máquina com o sistema operacional *Ubuntu - 11.04 versão 32 bits* feitas pelo autor deste texto, e os gols são mostrados na Tabela 3.1, Tabela 3.2 e Tabela 3.3 a seguir:

Tabela 3.1: *Uva TriLearn 2003 versus Helios_base*

	UvA TriLearn 2003	*Helios_base*
Jogo 1	0	20
Jogo 2	0	25
Jogo 3	0	17
Jogo 4	1	21
Jogo 5	0	19
Total	**1**	**102**

Tabela 3.2: *Uva TriLearn 2003 versus WrightEagle2D_BASE*

	UvA TriLearn 2003	*WrightEagle2D_BASE*
Jogo 1	5	6
Jogo 2	10	4
Jogo 3	2	4
Jogo 4	7	5
Jogo 5	5	6
Total	**29**	**25**

Tabela 3.3: *WrightEagle2D_BASE versus Helios_base*

	WrightEagle2D_BASE	*Helios_base*
Jogo 1	0	24
Jogo 2	0	25
Jogo 3	0	21
Jogo 4	0	24
Jogo 5	0	21
Total	**0**	**115**

O quadro geral de resultados é mostrado na Tabela 4.3.

Tabela 3.4: Quadro geral das partidas

Time-Base	Gols Marcados	Gols Sofridos	Vitórias	Derrotas
WrightEagle2D_BASE	25	144	3	7
UvA TriLearn 2003	30	127	2	8
Helios_base	217	1	10	0

A avaliação de desempenho feita pelo autor deste texto não ocorreu somente no resultado das partidas, mas em um conjunto de atributos pertinentes a uma partida de futebol como: poder de marcação, efetividade no ataque, quantidade de situações de gols criadas, gasto energético dos jogadores. Neste conjunto de observações o time com melhor desempenho foi o *Helios_base*, sendo este o escolhido para este estudo.

Mesmo com a elasticidade em todos os placares, o time *Helios_base* mostrou algumas deficiências graves como o gasto energético dos jogadores que a cada final de turno os jogadores não conseguiam correr mais, apenas caminhavam lentamente em campo. Outra deficiência notada foi a falta de variedade de jogadas realizadas pelo time, que na maioria das vezes realiza a mesma jogada que é chegar à linha de fundo e cruzar a bola para o companheiro atrás.

3.6 Comentários Finais

Neste capítulo foi apresentado um comparativo entre os times-base, sendo analisados os códigos, facilidade de instalação, facilidade de alteração, documentação, análise de jogadas, análise de resultados, entre outras.

Na sequência será mostrada as alterações realizadas no time UFLA2D, com implementações de novas técnicas, melhorias nos códigos, análise de desempenho das alterações e análise de resultados de partidas.

4 MODIFICAÇÕES REALIZADAS NO UFLA2D

A seguir serão detalhadas as mudanças promovidas no time UFLA2D a fim de corrigir falhas, melhorar decisões e/ou aspectos e suprir necessidades encontradas. Tais alterações foram embasadas na leitura de trabalhos de outras equipes, correções apontadas pelos desenvolvedores do time *Helios_base* ou observações feitas em inúmeras partidas criadas por este autor.

4.1 Economia de Energia

Uma das principais falhas encontrada no time *Helios_base* foi o gasto energético sofrido pelos jogadores no decorrer das partidas. Pois, próximo ao apito final do árbitro vários deles encontram-se quase sem energia para movimentar-se, com isso há uma notável queda de rendimento do time.

A seguir, mostra-se parte do código-fonte já alterado e com alguns comentários, no qual são definidas parte das estratégias adotadas no decorrer dos jogos, o arquivo *strategy.cpp*.

```
/* Utilizando a energia economizada */
else if ((2300 < wm.time().cycle() && wm.time().cycle() < 3000)
      || (5000 < wm.time().cycle() && wm.time().cycle() < 6000))
{
   dash_power = ServerParam::i().maxDashPower();
   dlog.addText( Logger::TEAM,
               __FILE__": (get_normal_dash_power) time is overing",
               dash_power );
   return dash_power;
}

/* Companheiro em boa posicao */
```

```
else if ( wm.existKickableTeammate()
          && wm.ball().distFromSelf() < 20.0 )
{
    dash_power = std::min( my_inc * 0.8,
                           ServerParam::i().maxDashPower() );
    dlog.addText( Logger::TEAM,
                  __FILE__": (get_normal_dash_power)
                  exist kickable teammate. dash_power=%.1f",
                  dash_power );
}

/* Posicao de impedimento */
else if ( wm.self().pos().x > wm.offsideLineX() )
{
    dash_power = ServerParam::i().maxDashPower()*0.95;
    dlog.addText( Logger::TEAM,
                  __FILE__": in offside area. dash_power=%.1f",
                  dash_power );
}

/* Situacao normal */
else
{
    dash_power = std::min( my_inc * 1.4,
                           ServerParam::i().maxDashPower() );
    dlog.addText( Logger::TEAM,
                  __FILE__": (get_normal_dash_power)
                  normal mode dash_power=%.1f",
                  dash_power );

}

return dash_power;
```

Cada agente inicia os tempos de jogo com 8000 unidades de energia, sendo que seu gasto máximo permitido por ciclo é de 100 unidades de energia.

Nas linhas 2 e 3 o agente captura a condição do final dos turnos, sendo observado o tempo de jogo. Neste intervalo de tempo para todas as ações do agente é injetado o máximo de energia.

A primeira condição, definida nas linhas 3 e 4 visa alertar sobre uma situação de risco, ou seja, a posição da bola está próxima a linha do gol aliado. Nesse caso é utilizada total energia permitida, para interceptar a bola.

Se o agente estiver em um estado de recuperação (linha 22), definido previamente pelo servidor *RCSS* será utilizada uma quantidade de energia podendo variar entre 0 e 10.

Caso exista um companheiro próximo (linhas 31 e 32) ao detentor da bola (menor que 20 metros de distância na escala do jogo) e com boas condições, ou seja, pouca marcação e/ou próximo ao gol adversário o gasto energético será administrado de forma a economizar.

Na situação em que o agente está em posição de impedimento (linha 41) ele utilizará 95% de sua energia total de arranque para mover-se até uma posição de não impedimento. Economia de 5% no gasto energético para realizar esta ação.

As linhas 49, 50, 51 e 52 definem quando o adversário está com o domínio da bola, caso onde não foi alterado o código.

Com a situação normal de jogo, que são todas as outras situações diferentes das citadas acima, pode ser explorada uma maior economia de energia, pois, corresponde a maior parte do tempo e o intervalo do valor de gasto energético gira entre 52 e 67. Essas alterações permitiram uma boa economia energética, comprovada após os testes feitos em partidas contra o time-base (*agent2D*) original.

As modificações foram feitas escolhendo valores empiricamente após a realização de 15 partidas contra o time original. Para utilizar a energia economizada no decorrer da partida foi criada uma regra que analisa o tempo de jogo, sendo focado o final de cada tempo visando aproveitar o desgaste físico dos atletas adversários.

Os agentes seguem um comportamento padrão para se movimentarem e a quantidade de energia aplicada vai depender de sua situação atual, podendo variar de acordo com os aspectos mostrado na Tabela 4.1:

Tabela 4.1: Gastos Energéticos Padrão

Situação Atual	Quantidade de Energia Aplicada
Companheiro em boa posição	[40,50]
Estado de recuperação	[0,10]
Posição de impedimento	100
Posse de bola do adversário	[10,100]
Situação de risco	100
Situação normal	[68,85]

Na Tabela 4.1, a situação atual é representada pelas seguintes características:

- **Companheiro em boa posição:** se existe um companheiro em uma posição melhor próximo ao agente que detém a bola, então não há necessidade de se aplicar grande quantidade de energia.
- **Estado de recuperação:** se sua energia disponível for menor que 2000 (número padrão para as regras atuais). Não alterado no código do time-base, pois, sua energia já estava muito baixa.
- **Posição de impedimento:** se o jogador está em impedimento. Diminuição de 5% na energia utilizada para o jogador mover-se para uma posição legal.

- **Posse de bola do adversário:** se a bola está sob posse do adversário, inalterado no código do time-base.
- **Situação de risco:** se a bola estiver em uma zona perigosa, inalterado no código do time-base.
- **Situação normal:** outras situações que diferem das citadas acima e é a maior parte do tempo de jogo, sendo possível obter uma economia maior.

Na Tabela 4.2 são mostrados os gastos energéticos agora com suas devidas alterações nos valores, que visou poupar energia durante a partida.

Tabela 4.2: Gastos Energéticos Modificados

Situação Atual	Quantidade de Energia Aplicada
Companheiro em boa posição	[32,39]
Estado de recuperação	[0,10]
Posição de impedimento	95
Posse de bola do adversário	[10,100]
Situação de risco	100
Situação normal	[52,67]

Os resultados observados após essas modificações representam uma significante melhora no final dos turnos, onde são aplicados as economias energéticas. Estes números demonstram assim um grande potencial nessa área do futebol para a aplicação de técnicas de inteligência artificial e otimização a fim de explorar e maximar os gastos energéticos dos atletas aplicando-os em intervalos estratégicos na partida.

Na Tabela 4.3 são mostrados os resultados isolados da melhoria nos gastos energéticos e sua aplicação no final de cada tempo da partida, ou seja, entre o

intervalo de 2300 ciclos até 3000 ciclos e 5000 ciclos até 6000 ciclos, obtidos através da realização de dez jogos contra o time *Helios_base*.

Tabela 4.3: Gols marcados ao final de cada tempo

Jogo	Sem alteração	Com alteração
Jogo 1	0	4
Jogo 2	1	5
Jogo 3	0	3
Jogo 4	0	2
Jogo 5	1	2
Jogo 6	0	3
Jogo 7	2	3
Jogo 8	0	1
Jogo 9	0	0
Jogo 10	0	2
Total	**4**	**25**

4.2 Marcação

A característica padrão de marcação utilizada é conhecida por marcação por zona, onde os marcadores ficam apenas 'cercando' seu adversário. Essa é uma escolha perigosa, pois, dá-se espaço para o adversário trabalhar com a bola.

A análise feita a cada momento do jogo em que o time não detém a bola é feita calculando a probabilidade de sucesso tanto na tentativa de interceptar o adversário de forma limpa como na interceptação com falta. Com base nesses dados, foram

feitas alterações nos parâmetros desses cálculos diminuindo a área de atuação e aumentando a precisão nos cálculos.

Outra alteração realizada foi a criação de uma situação de roubada de bola quando há uma situação eminente de risco, onde é verificado se existe um jogador adversário em uma boa condição de chute, a bola está em direção ao gol. Ao detectar uma dessas situações é disparada uma ação para tentar roubar a bola sem cometer falta.

Com a realização de 15 partidas de teste foi constatado uma melhora no desempenho da tentativa de roubar a bola. Os agentes ficam melhores posicionados, diminuíram a quantidade de tentativas de interceptação e aumentaram a eficácia. Com o time marcando melhor, existe a tendência na diminuição dos gols sofridos e uma maior posse de bola. Em média o time-base realiza 43 tentativas de roubadas de bola e com a alteração passou a ter uma média superior a 120 tentativas de roubadas de bola.

O arquivo que define o comportamento básico de interceptação da bola é chamado de *bhv_basic_tackle.cpp*, Figura 4.1. Nele foi inserido a execução de uma ação conhecida por *Body_Intercept2009*[1], porque o número de tentativas de interceptações da bola era baixa, média de 43 por partida e com isso afetava diretamente na quantidade de gols sofridos.

[1]Responsável por elaborar um plano de interceptação eficiente

```
bool ball_will_be_in_our_goal = false;

if ( self_reach_point.x < -SP.pitchHalfLength() )
{
    const Ray2D ball_ray( wm.ball().pos(), wm.ball().vel().th() );
    const Line2D goal_line( Vector2D( -SP.pitchHalfLength(), 10.0 ),
                            Vector2D( -SP.pitchHalfLength(), -10.0 ) );

    const Vector2D intersect = ball_ray.intersection( goal_line );
    if ( intersect.isValid()
       && intersect.absY() < SP.goalHalfWidth() + 1.0 )
    {
         ball_will_be_in_our_goal = true;

         dlog.addText( Logger::TEAM,
                       __FILE__": ball will be in our goal.
                       intersect=(%.2f %.2f)",
                       intersect.x, intersect.y );
    }
}

if ( wm.existKickableOpponent()
    || ball_will_be_in_our_goal
    || ( opp_min < self_min - 3
       && opp_min < mate_min - 3 )
    || ( self_min >= 5
       && wm.ball().pos().dist2( SP.theirTeamGoalPos() )
       < std::pow( 10.0, 2 )
       && ( ( SP.theirTeamGoalPos() - wm.self().pos() ).th()
       - wm.self().body() ).abs() < 45.0 )
   )
{
   Body_Intercept2009().execute( agent );
}
```

Figura 4.1: Arquivo *bhv_basic_tackle.cpp* modificado

A ação de tentar interceptar a bola de forma satisfatória (linha 32) é executada quando a situação atual se encaixa dentre as especificadas entre as linhas 22 e 29 demonstrado no código acima. Com esta alteração pode ser notado um aumento na quantidade de roubadas de bola e de forma eficiente, ou seja, sem cometer falta.

Para um melhor aproveitamento desta descoberta, pode ser implementada uma otimização na tentativa de interceptação da bola cometendo ou não falta no adversário. Como por exemplo, utilizar-se da técnica de Inteligência Artificial bastante utilizada no domínio da simulação do futebol, conhecida por *Q-Learning* (NERI; SANTOS; FABRO, 2011), sendo uma ramificação do **Aprendizado por Reforço**.

5 RESULTADOS E DISCUSSÃO

Este capítulo apresentará os resultados dos testes realizados com o time-base *Helios_base* alterado, agora podendo ser chamado de **UFLA2D**. Alterações estas, que foram descritas no capítulo anterior surtiram efeitos tanto positivos quanto negativos.

As partidas foram realizadas seguindo os padrões adotados nos eventos realizados pela *RoboCup*. Especificamente foi adotado como sistema operacional o *Ubuntu* 12.04 LTS 64 bits, *rcssserver* versão 15.0.1 e *rcssmonitor* versão 15.0.0, sendo todos estes sistemas disponibilizados gratuitamente.

As partidas foram realizadas de forma local, ou seja, o servidor e os dois times eram rodados na mesma máquina, que tem um processador *Intel® Core™ i3*, com 2 GB de memória *RAM* e com 500 GB de disco rígido.

Os binários dos times usados nos testes são disponibilizados gratuitamente pelos organizadores da competição através de um repositório de dados (ROBOCUPFTP, 2012).

5.1 *OXSY*

O time romeno *Oxsy* foi criado em Julho de 2002, com o projeto de sistemas multiagente de Sebastian Marian (OXSY, 2012). Teve como a primeira participação na *RoboCup* no ano de 2003, com a colocação do 18º lugar. No ano de 2011 apresentou excelentes atuações, ficando em 4º lugar no mundial. A Tabela 5.1 apresenta os resultados dos jogos realizados contra este time.

Tabela 5.1: Primeira bateria de jogos

UFLA2D	***OXSY***
1	0
2	1
1	3
1	2
3	2
1	5
0	4
0	0
0	3
1	2
10	**22**

Os resultados acima mostram um bom desempenho do time UFLA2D, que apresentou um bom futebol com passes bem executados, realização de jogadas trabalhadas (Ex.: lançamentos em profundidades) e um bom saldo de gols mesmo tendo perdido mais do que ganhado, pois, o adversário é detentor de uma excelente colocação no último mundial (2011).

5.2 *RaiC*

O time japonês *RaiC* da Universidade Nacional de Tecnologia *FUKUI*, situada na cidade *Geshi Sabae* (RAIC, 2012). Em 2011 conseguiu a 8º colocação no mundial. A Tabela 5.2 detêm os resultados das partidas.

Tabela 5.2: Segunda bateria de jogos

UFLA2D	***RaiC2011***
3	4
8	2
4	3
3	5
4	4
3	1
0	4
3	2
6	1
3	1
37	**27**

Os jogos realizados contra o time *RaiC2011* mostraram um UFLA2D bem organizado, equilibrado e que na maioria das partidas soube aproveitar bem os momentos de final de turno, onde, utilizou-se a energia armazenada no decorrer do jogo em seu favor. Vários gols foram marcados no final dos turnos, com isso o UFLA2D conseguiu uma vantagem sobre seu adversário.

5.3 *Ri-one*

O time japonês *Ri-one* é alocado na Faculdade de Ciência e Tecnologia, situada na cidade *Ritsumeikan* (RI-ONE, 2012). Este time conta com várias atividades e participações na Liga Simulada da *RoboCup*. Eles objetivam a melhoria de competên-

cias por lições de aprendizagem e atividades inter-curriculares e extra-curriculares no segmento de pesquisa proporcionado pelo futebol de robôs.

O Ri-one obteve o 13º lugar no mundial em 2011. A Tabela 5.3 detêm os resultados das partidas.

Tabela 5.3: Terceira bateria de testes

UFLA2D	***Ri-one***
5	0
5	1
11	1
13	1
10	3
4	0
8	0
5	1
10	1
6	0
77	**8**

O time UFLA2D teve um desempenho notável contra o time japonês que é detentor de grande experiência em campeonatos e eventos organizados pela *RoboCup*, mostrando-se um time bastante agressivo no ataque e bem equilibrado defensivamente.

5.4 *Helios2011*

HELIOS da Universidade Fukuoka, AIST no Japão participa das competições em duas dimensões desde 2000 (AKIYAMA *et al.*, 2011). Em 2011 obteve a segunda colocação no campeonato mundial.

Tabela 5.4: Quarta bateria de testes

UFLA2D	***Helios2011***
0	6
0	4
0	5
0	3
1	2
2	1
0	1
1	3
1	2
0	3
5	**30**

Os jogos contra o time *Helios2011* foram em sua maioria bem pareados, tendo disputas em todos os metros quadrados do campo, com forte marcação e belas jogadas.

5.5 Discussão

As partidas foram de suma importância para a realização deste trabalho, já que permite mensurar as alterações feitas, expor falhas e pontos positivos, análise individual e coletiva dos agentes e a validação do trabalho. Como um todo, os jogos da equipe UFLA2D demonstraram-se bastante satisfatórios, mesmo com algumas fragilidades expostas. Fazendo uma breve previsão de resultados (coisa nem sempre correta no mundo futebolístico), pode-se dizer que o time UFLA2D se tivesse disputado o mundial no ano de 2011 realizado em Istambul obteria uma colocação próxima à 6ª.

As falhas notadas se dão em todos os níveis do time, tanto nas tarefas simples como aproximar jogadores na hora de cobrança de escanteios e laterais quanto nas tarefas mais complexas como voltar a bola para o campo de defesa quando não há jogadas e/ou jogadores disponíveis para se criar boas oportunidades.

Quanto às alterações que surtiram efeitos positivos, como o aproveitamento da energia e a criação de situações de jogo em que se deve tentar interceptar a bola, mostraram-se com grande potencial de melhoria se sujeitadas à técnicas de Inteligência Computacional e/ou Otimização em sistemas multi-agentes.

Outro aspecto que deve ser levado em consideração é a melhoria do algoritmo de *Action Chain* (ver subseção 3.3), onde é depositada todas as ações inteligentes em cadeia que vislumbram sempre um objetivo, como por exemplo fazer gol. Tal cadeia, não consegue recuperar-se de eventos que não foram previamente mapeados, ou não se adaptam de forma correta com situações extremas. Notado que há algumas situações de jogo onde o agente não sabe o que fazer com a bola e fica parado ou andando em círculos sem nenhuma perspectiva.

Para contornar tais fatos, pode-se criar novos estados a cada ação da cadeia fomentando criar amarrações para diversas situações da partida. Também a criação

de estados adaptativos, a fim de usar da criatividade dos agentes para se adaptarem à novas situações, como exemplo, a alteração no comportamento do time adversário.

6 CONCLUSÃO

Ao ser adotado o futebol como meio de pesquisa para a inteligência computacional, têm-se uma exploração muito grande das técnicas e métodos para autonomia e cooperação de agentes não humanos, pois, o ambiente futebolístico tem um grau elevado de dificuldade e aleatoriedade. Por ter aspectos bem semelhantes, os conhecimentos do futebol real podem ser aplicados na simulação, assim como os conhecimentos do futebol de robôs possa ajudar em um futuro próximo o futebol com humanos.

Desta forma, este trabalho apontou melhorias para o desenvolvimento de um sistema multi agente forte, rápido e equilibrado. Por meio da realização de partidas de teste, estudo do código e conhecimentos empíricos do futebol alcançou-se um bom toque de bola, precisão nos chutes, forte marcação, escolha correta das melhores jogadas, economia de gasto energético. Neste texto, foram destacadas as melhorias necessárias que possam embasar a aplicação de técnicas de inteligência computacional e otimização, com o intuito na melhora da tomada de decisão com ou sem a bola, individual ou coletivamente.

Ao término da bateria de testes para a escolha do time básico mais completo, ficou claro que o time com melhor desempenho em confrontos diretos foi o HELIOS_base. Em relação à disponibilidade de documentação, comentários no código-fonte e trabalhos científicos publicados, o time *UvA TriLearn* levou vantagem. Já em relação ao desempenho da equipe fornecedora do time-base, a equipe *WrightEagle Base* leva vantagem, pois, nas últimas quatro edições do campeonato mundial chegou à grande final conquistando a taça em três oportunidades. A Figura 6.1 demonstra algumas características importantes para a escolha do time-base.

Ação \ Time-base	*Helios_base*	*UvA TriLearn 2003*	*WrightEagle2D_ BASE*
Poder de Marcação	✓		
Efetividade no ataque	✓		
Quantidade de situações de gol criadas	✓		
Confronto direto	✓		
Gasto energético dos jogadores		✓	
Modularização do código fonte			✓
Facilidade de instalação		✓	
Documentação		✓	
Facilidade de alteração		✓	

Figura 6.1: Tabela comparativa entre os times-base

Após a adoção de um novo time-base para a equipe UFLA2D e com ajustes em algumas características, foram obtidos bons resultados em relação ao time antigo da equipe UFLA2D nos testes realizados contra equipes conceituadas no cenário mundial. Com base nas simulações das partidas, se a equipe UFLA2D tivesse participado do campeonato mundial no ano de 2011 (*RoboCup 2011 Istanbul - Turkey*) teria obtido uma colocação entre os 10 melhores times.

Como propostas de continuidade, pode ser acoplado a este trabalho meios da inteligência artificial para tentar melhorar o desempenho do time em aspectos apontados por este autor, assim como outros tantos aspectos disponíveis para alterações, pesquisas, inovações e melhorias.

O gasto energético é uma característica que pode ser explorada por técnicas de inteligência computacional, podendo ser aproveitado os relatos descritos nos capítulos anteriores. A estratégia de economia de energia de cada atleta durante

boa parte das partidas aproveitando-a ao final dos turnos de cada jogo pode ser melhorada através de uma modelagem *Fuzzy*.

A marcação dos jogadores é outro aspecto que pode ser melhorado em uma continuidade deste trabalho. O aprendizado por reforço pode ser usado para tentar melhorar o poder de marcação do time, atribuindo penalidades nas tentativas de roubadas de bola sem sucesso ou com falta.

Algoritmos adaptativos podem ser usados para maximizar a eficiência da equipe em relação à equipe adversária. Este aspecto pode ser explorado em relação ao placar e tempo da partida, que caso a equipe esteja vencendo e o jogo caminhando para o final os agentes podem segurar mais a bola em sua posse, podem dar passes curtos e constantes ao invés de arriscar jogadas mais difíceis e o atraso dos jogadores de frente para ajudar na marcação. Outro ponto que pode ser explorado é a adaptação em relação à estratégia da equipe adversária.

A criação de uma equipe de futebol de robôs com integrantes de diversos cursos da Universidade Federal de Lavras passa a ser vislumbrada como uma proposta de continuidade, já que a simulação em duas dimensões do futebol de robôs pode ser considerada como a porta de entrada para as demais competições da *RoboCup*. Este trabalho pode ser aproveitado como alicerce para estudos e pesquisas tanto na liga simulada quanto na liga física.

REFERÊNCIAS BIBLIOGRÁFICAS

2D, W. E. *Wright Eagle 2D. Visitado no dia 07 de Março de 2012*. 2012. Disponível em: <http://wrighteagle.org/2D/>.

AKIYAMA, H.; SHIMORA, H.; NAKASHIMA, T.; NARIMOTO, Y.; OKAYAMA, T. Helios2011 team description. RoboCup, v. 1, n. 1, p. 1–6, 2011.

ALCANTARA, A. R. de; SILVA, P. H. G. e. Ufla2d 2011: Team description paper. Competição Brasileira de Robótica, v. 1, n. 1, p. 1–3, 2011.

ALDEBRAN. *Aldebran Robotics Corporate Site, Visitado no dia 26 de Setembro de 2011*. 2011. Disponível em: <http://www.aldebaran-robotics.com/en-/Discover-NAO/Key-Features/hardware-platform.html>.

BOER, R. de; KOK, J. The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team. *Master's thesis, University of Amsterdam*, Citeseer, p. 1–133, 2002.

CHEN, M.; DORER, K.; FOROUGHI, E.; HEINTZ, F.; HUANG, Z.; KAPETANAKIS, S.; KOSTIADIS, K.; KUMMENEJE, J.; MURRAY, J.; NODA, I.; OBST, O.; RILEY, P.; STEFFENS, T.; WANG, Y.; YIN, X. *Users Manual: RoboCup Soccer Server — for Soccer Server Version 7.07 and Later*. [S.l.], February 2003. Disponível em: <http://helios.hampshire.edu/jdavila/cs278%-/virtual_worlds/robocup_manual-20030211.pdf>.

DESIGN, V. S. *RoboCup silver medal: Soccer robots from Eindhoven University*. 2011. Disponível em: <http://www.vision-systems.com/articles/2009/08/vc-cameras-support-winners-at-this-years-robocup.html>.

FRACCAROLI, E. Análise de desempenho de algoritmos evolutivos no domínio do futebol de robôs. Biblioteca Digital de Teses e Dissertações da USP, 2011.

JR, L. C.; BIANCHI, R.; MATSUURA, J. Aprendizado por reforço acelerado por heurísticas no domínio do futebol de robôs simulado. v. 1, n. 1, p. 1–6, 2011.

MACKWORTH, A. On seeing robots. World Scientific Press, Singapore, p. 1–13, 1993.

NERI, J.; SANTOS, C.; FABRO, J. Descrição do time gpr-2d 2011. Competição Brasileira de Robótica, v. 1, n. 1, p. 1–4, 2011.

OXSY. *OXSY. Visitado no dia 09 de Março de 2012*. 2012. Disponível em: <http://www.oxsy.ro/blog/>.

RAIC. *RAIC2011. Visitado no dia 10 de Março de 2012*. 2012. Disponível em: <http://www.fukui-net.ac.jp/˜tomomi>.

REIS, L. Coordenação em sistemas multi-agente: Aplicações na gestão universitária e futebol robótico. Faculdade de Engenharia da Universidade do Porto, 2003.

RI-ONE. *Ri-one. Visitado no dia 07 de Março de 2012*. 2012. Disponível em: <http://www.pj.is.ritsumei.ac.jp/ri-one/>.

ROBOCUP. *RoboCup Official Site, Visitado no dia 23 de Setembro de 2011*. 2011. Disponível em: <http://www.robocup.org/about-robocup/a-brief-history-of-robocup/>.

ROBOCUPFTP. *RoboCup FTP. Visitado no dia 15 de Maio de 2012*. 2012. Disponível em: <http://www.socsim.robocup.org/files/>.

ROBOCUPGALLERY. *RoboCup Gallery. Visitado no dia 15 de Maio de 2012*. 2011. Disponível em: <http://www.robocup2010.org/home_Gallery.php>.

ROBOCUPPROJECT. *The RoboCup Soccer Simulator. Visitado no dia 15 de Maio de 2012*. 2012. Disponível em: <http://sourceforge.net/projects/sserver/>.

SILVA, H. da L.; SIMÕES, M.; ARAGÃO, H. Desenvolvimento de controladores fuzzy para robôs jogadores de futebol simulado do tipo atacante. *Workshop de Trabalhos de Iniciação Científica e Graduação Bahia-Alagoas-Sergipe. Anais da 7a Escola Regional de Computação Bahia-Alagoas-Sergipe.*, Porto Alegre: SBC, 2007.

SIMSPARK, W. *SimSpark. Visitado no dia 08 de Março de 2012*. 2012. Disponível em: <http://en.wikipedia.org/wiki/SimSpark>.

SIMULATION, R. S. *RoboCup 3D Soccer Simulation League*. 2012. Disponível em: <wiki.robocup.org/wiki/Soccer_Simulation_League>.

TEAM, H. *RoboCup Tools Download. Visitado no dia 28 de Outubro de 2012*. 2012. Disponível em: <http://pt.sourceforge.jp/projects/rctools%-/releases/?package_id=4887>.

WIKIPEDIA, t. f. e. F. *RoboCup-3D-Soccer-Field. Visitado no dia 23 de Junho de 2012*. 2012. Disponível em: <http://en.wikipedia.org/wiki/File:RoboCup-3D-Soccer-Field.jpg>.