FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Algoritmos para Detecção Automática de Eventos aplicados ao Futebol Robótico Simulado

José Maria Rosa de Sousa de Mendonça e Moura

Relatório de Projecto/Dissertação

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Luis Paulo Reis (Professor Doutor)

Co-orientador: Pedro Abreu (Engenheiro)

28 de Junho de 2009

Algoritmos para Detecção Automática de Eventos aplicados ao Futebol Robótico Simulado

José Maria Rosa de Sousa de Mendonça e Moura

Relatório de Projecto/Dissertação

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Rosaldo Rossetti (Professor Doutor)

Arguente: César Analide (Professor Doutor)

Vogal: Luis Paulo Reis (Professor Doutor)

31 de Julho de 2009

Resumo

O futebol é um desporto colectivo, disputado por duas equipas de onze jogadores cada, cujo objectivo é marcar mais golos que a equipa adversária. O sucesso do futebol a nível mundial não tem precedentes, em todos os continentes as pessoas vibram no apoio ao seu clube. Este sucesso faz com que os campeonatos mais vistos pelo mundo fora, como é o caso do campeonato Inglês, gerem receitas enormes. Os exemplos mais comuns são o caso do clube inglês Manchester United e o Real Madrid, de Espanha, que todos os anos, antes do início da temporada, efectuam digressões pelo continente asiático e americano de maneira a gerarem mais receitas e a satisfazerem os seus adeptos espalhados pelo mundo fora. Devido ao crescimento do futebol, do ponto de vista económico, ganhar competições, hoje mais que nunca, envolve não apenas ganhar prestigío mas também permite garantir uma maior fonte de receitas. Por essa razão, cada vez mais os clubes procuram na tecnologia uma forma de encontrar uma mais valia, na compreensão do adversário, de forma a atingirem os resultados pretendidos.

Esta dissertação consistiu no desenvolvimento de algoritmos para a detecção de conjunto de eventos pré-definidos, num jogo de futebol. Devido às dificuldade em aceder aos dados de jogos de futebol real, optou-se pela utilização de dados proveniente de uma competição de futebol virtual: Robocup. O Robocup é uma competição de futebol robótico, onde as equipas são formadas por robôs programados por investigadores de todo o mundo, cujo objectivo é expandir e desenvolver métodos de inteligência artificial. Uma das vertentes do Robocup é a liga de simulação, onde cada equipa, virtual, sem a presença de robôs, consiste num conjunto de agentes que cooperam entre si de forma a ganhar um jogo perante uma outra equipa nas mesmas condições. Foram utilizados os dados da "liga de simulação"devido ao elevado número de características comuns com os jogos de futebol real. Os algoritmos foram desenvolvidos tendo como base, dados que consistiam nas posições físicas dos agente em cada momento de jogo, isto é, os dados consistiam na posição de cada jogador e da bola em cada instante de tempo. Devido à subjectividade associada a cada evento, desenvolveu-se uma plataforma de suporte ao desenvolvimento de detectores de eventos, de forma a aumentar a flexibilidade de cada detector, podendo, cada algoritmo, ser alterado e configurado facilmente. Os detectores de eventos apresentam taxas de sucesso bastante diferentes: os eventos facilmente formalizados como, por exemplo, o passe ou o offside apresentam taxas de detecção acima dos 90%, por outro lado, eventos, como por exemplo, um remate, apresentam taxas de sucesso um pouco mais baixas devido à subjectividade associada a cada um deles.

Abstract

Soccer is a team game played by two teams of eleven players each, whose goal is to introduce the ball in the other's team goal. The success of soccer in the whole world has no match and in all continents people support enthusiastically their teams. This success allows most known leagues, like "barclays premier league", to generate millions. The most striking successes are teams like Manchester United or Real Madrid, that every year before a season starts, make tours through Asia or America that give them millions in revenue and please their legions of supporters. Due to its growth, under an economic standpoint, to win a competition not only gives more fame but also more millions. Therefore teams look to technology in order to gain advantage in understanding their opponent strategies so they can obtain the desired results.

This work consisted in developing algorithms to detect a set of predefined events in a soccer game. Due to difficulties in obtaining data from a real game we decided to use data from a league of virtual soccer: Robocup. Robocup is a league of robotic soccer, where the teams are formed by robots programmed by researchers of the whole word whose aim is to develop methods of artificial intelligence. One of types of Robocup is the simulation league, where a virtual team , without the presence of robots, is made of a set of agents that cooperate among themselves in order to win a game opposing a similar team. We used data from games of this league due to its similarity with soccer games. The developed algorithms were based on data that consisted in the position of the players and the ball in every instant of time. Since each event is ambiguous because sometimes we cannot decide which type of event it is, we developed a framework to support the detectors of events so we could increase the flexibility of each detector allowing each algorithm to be easily changed. The event detectors present a success rate different based on the complexity of the events themselves; a pass or an off-side present success rate above 90% but a Shoot presented success rates below that due to the difficulty in classifying it as a kick or as pass, for example.

Agradecimentos

Agradeço ao meu orientador, Prof. Luis Paulo Reis, pela sua disponibilidade, pelos seus esclarecimentos e pelo seu suporte na construção deste documento.

Agradeço ao meu co-orientador, Eng.º Pedro Abreu pelo seu suporte, ajuda, compreensão, confiança e companheirismo. Pela atenção que dedicou ao meu trabalho, pela consideração das minhas ideias e pela motivação que me causou. Agradeço-lhe pelos inúmeros dias em que discutimos inúmeros factos sobre futebol sem chegar a conclusão alguma.

À minha família por me terem sempre ajudado e acreditado em mim.

José Maria Mendonça e Moura

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Abreviaturas e Símbolos

DSL	Domain Specific Language
IA	Inteligência Artificial
JVM	Java Virtual Machine
GUI	Graphical User Interface
API	Application Programming Interface
MVC	Model View Controller
LOTG	Laws Of The Game
WWW	*World Wide Web*

Capítulo 1

Introdução

1.1 Contexto/Enquadramento

Inserido nos Jogos Desportivos Colectivos, o futebol é um jogo onde duas equipas, constituídas por 11 jogadores disputam uma partida. O futebol é um desporto colectivo, disputado por duas equipas com onze jogadores cada, uma bola e regulado por 2 fiscais-de-linha e um árbitro, unanimemente considerado o desporto mais popular do mundo [Dun99]. O objectivo do jogo é bastante simples: consiste, durante a duração de um jogo, em fazer mais golos do que a equipa adversária. Um golo é definido como a colocação da bola dentro da baliza da equipa adversária.

O futebol actual tem as suas raízes em Inglaterra, onde em 1863 [Wika], foi formada a "The Football Association"que estabeleu as regras básicas do jogo, que ainda hoje continuam em vigor.

As leis básicas do futebol (também conhecidas pela sigla LOTG), estabelecidas em 1983, são 17 e consistem nas seguintes [Wikb]:

1. Tamanho do campo do jogo
2. Propriedades da bola
3. Número de jogadores de cada equipa (no máximo onze jogadores, em que um jogador é guarda-redes)
4. O equipamento de cada equipa
5. O árbitro responsável por certificar que cada equipa cumpre as regras
6. Os assistentes do árbitro
7. Duração do jogo

8. Definição do começo e re-começo do jogo
9. Definição das posições da bola, dentro e fora do campo
10. Golo
11. Lei do fora-de-jogo
12. Faltas e conduta obrigatória de cada jogador
13. Definição de livres
14. Definição de pontapés de grande penalidade
15. Lançamento de linha lateral
16. Definição de "pontapé de baliza"
17. Definição de "pontapé de canto"

Actualmente as leis do jogo são publicadas e alteradas pela International Football Association Board (IFAB) e pela FIFA (International Federation of Association Football) organização responsável pela gestão do desporto a nível internacional, bem como, da organização de eventos desportivos, com destaque para o FIFA World Cup. É também importante salientar que as regras, segundo a FIFA, estão sujeitas a uma certa flexibilidade a quando da sua aplicação, dependendo das situações em causa.

O sucesso e a popularidade do futebol não têm precedentes, considerado o desporto rei, movimenta pessoas, organizações e indústrias no mundo inteiro. Pessoas de países e raças diferentes unem-se através de um desporto, o que obriga os clubes a deslocarem-se a outros continentes só para agradarem aos seus fãns. É hoje natural, que um clube europeu de sucesso, faça digressões pelo continente asiático ou pela América do Norte, de maneira a aumentar as suas receitas, o futebol é hoje uma gigantesca indústria de entretenimento, onde, por exemplo, o campeonato do mundo de futebol é o evento com maior audiência do mundo, tendo sensivelmente o dobro dos espectadores dos Jogos Olímpicos. [Cora]

Devido ao sucesso e ao dinheiro envolvido no futebol, várias indústrias cresceram à sua volta, as equipas mais que nunca, tem de vencer os torneios em que estão envolvidos pois os "prize money"são cada vez maiores, os programas de televisão sobre o desporto são cada vez em maior número, os blogs, vídeos e imagens dos clubes proliferam na internet.

Devido à alta competitividade, os detalhes num jogo, têm hoje, mais que nunca, uma importância vital para as equipas, podendo determinar o sucesso ou insucesso delas, daí o mercado necessitar de ferramentas que permitam uma melhor observação da equipa adversária de maneira a detectar pontos fortes a anular, e pontos fracos para explorar.

1.2 Projecto

Este projecto de mestrado visa o desenvolvimento de ferramentas e algoritmos que permitam, com o maior datalhe possível, extrair automaticamente informação de eventos ocorridos em jogos de futebol. Este projecto nasce da dificuldade e ineficiência dos sistemas existentes em efectuarem uma análise estatística de um jogo futebol.

Devido à própria natureza do jogo e à flexibilidade das regras, é bastante complexo, com precisão, uma pessoa ou um sistema informático detectar os eventos ocorridos num jogo. A grande dificuldade consiste em definir um evento formalmente. Existem eventos cuja definição depende de pessoa para pessoa, sendo muitas vezes a sua definição bastante ambígua. Um caso bastante comum é o remate, o remate é inúmeras vezes definido como um chuto na bola por parte de um jogador, da equipa atacante, com o objectivo de efectuar um golo. Não são raras as vezes em que um jogador, que pretende executar uma passe para outro jogador, efectua acidentalmente um golo, neste caso como seria classificado o evento? Como remate ou como passe mal efectuado? Estas nuances exigem por parte dos algoritmos desenvolvidos uma flexibilidade e facilidade de adaptação muito elevadas.

Se interpretarmos um jogo de futebol como um conjunto de entidades: jogadores e bola, em movimento, veremos a dificuldade, devido à presença de inúmeras variáveis externas, como o atrito causado pelo vento, em definir um determinado evento. Tomemos como exemplo um chuto: o chuto pode ser identificado com uma acelaração na bola, no entanto, se considerarmos a presença de atrito tal afirmação já não é válida, pois o vento também poderia provocar tal acelaração.

Era também necessário, na definição das regras dos eventos, que não existissem sobreposição de regras, isto é, a detecção de cada evento seria baseada num conjunto de regras que não iria interferir na definição de outros eventos. Com esta ideia em prática haveriam dois ganhos, um do ponto de vista da eficiência do sistema (o cálculo dos eventos poderia ser efectuado em paralelo) e o outro do ponto de vista da flexibilidade (a definição de um evento não afectaria nenhum outro).

Do ponto de vista funcional um sistema de detecção de eventos pode ser divido em 2 fases: aquisição e posterior análise dos dados . O objectivo deste projecto consistia no desenvolvimento de algoritmos de análise de dados, produzindo uma lista dos eventos detectados num jogo que permitissem a um treinador de futebol não só avaliar o desempenho da sua equipa como também do seu adversário. A fase de aquisição de dados resultaria da avaliação de sistemas já existentes.

1.3 Motivação e Objectivos

O projecto em causa tem como objectivo a extracção automática de eventos em jogos de futebol. Os eventos considerados são os eventos básicos de um jogo de futebol: pas-

ses certos, passes errados, remates, foras-de-jogo, etc no qual outras heurísticas podem operar. A ideia base consiste na definição de um conjunto de regras que operem sobre os dados de um determinado jogo, onde os dados incluem a posição de cada jogador e da bola em cada instante, de maneira a detectar os eventos nele ocorridos. De salientar que, com o objectivo de detectar os vários eventos que acontecem num jogo, serão usados apenas informações referentes às coordenadas dos jogadores e da bola ao longo do jogo.

Os objectivos do projecto consistiam em:

1. Análise dos métodos existentes de recolha de dados numéricos de jogos de futebol
2. Análise das soluções existentes de detecção automática de eventos
3. Definição do conjunto de eventos a detectar
4. Escolha de tecnologias que visem o desenvolvimento do protótipo de maneira rápida e flexível
5. Desenvolvimento e teste do protótipo
6. Comparação de resultados

A recolha de dados sobre o jogo constituíu um ponto crucial no desenvolvimento do projecto. A solução a encontrar tinha de demonstrar uma elevada precisão no fornecimento de dados, como tal, as seguintes abordagens foram consideradas:

- Fazer o tracking da posição de cada jogador e da bola, em cada instante, num conjunto elevado de jogos
- Utilizar dados, já extraídos por outra entidade

A primeira opção apesar de nos garantir a precisão de dados que pretendiamos (pois o controlo sobre o sistema seria total), era impraticável devido ao tempo disponível para o desenvolvimento do projecto, juntamente com o custo económico deste género de soluções . Neste sentido a segunda opção acabou por ser escolhida,contudo, devido à inflexibilidade das entidades possuidoras dessa informação (FIFA e Deltatre por exemplo) em fornecê-las, mesmo no âmbito académico, decidiu-se utilizar os dados disponíveis do RoboCup. O Robocup é uma competição internacional entre robôs(reais ou simulados) em jogos de futebol cujo objectivo é promover e desenvolver a inteligência artificial[Wikc]. Devido às semelhanças entre os jogos simulados no robocup e os jogos reais, juntamente com a disponibilidade dos dados, os dados dos jogos do robocup foram utilizados para o desenvolvimento do protótipo.

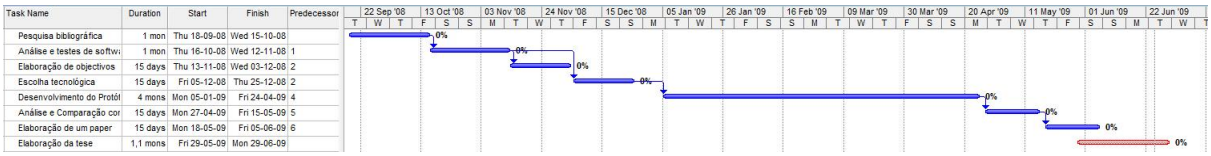

Figura 1.1: Diagrama de gant

1.4 Estrutura da Dissertação

Neste capítulo foi feita uma introdução ao projecto, apresentando a motivação e os objectivos para a sua realização, onde é também apresentado o plano de trabalhos. No segundo capítulo é apresentado o estado da arte em cada uma das componentes do projecto onde são discutidas as várias opções. No capítulo 3 são apresentadas e justificadas as escolhas tecnológicas. No quarto capítulo é feita uma apresentação do Robocup, explicando a sua estrutura e as diferentes competições que engloba. No quinto capítulo é discutido a implementação do protótipo onde são apresentados várias detalhes que se revelaram vitais para a obtenção dos resultados pretendidos. No capítulo 6 os resultados são apresentados e justificados, e, por fim, no sexto capítulo são apresentadas as conclusões bem como delineados futuros desenvolvimentos do projecto.

1.5 Plano de Trabalho

O plano do trabalho consistia nas seguintes fases:

1. Pesquisa bibliográfica
2. Análise e teste de software existente
3. Elaboração de objectivos
4. Escolha tecnológica
5. Desenvolvimento do protótipo
6. Análise e comparação de resultados
7. Elaboração de um paper
8. Escrita da tese de mestrado

Como ilustrado no diagrama de grant:

Capítulo 2

Revisão Bibliográfica

Neste capítulo é avaliado o "state of the art"em cada uma das fases do projecto:

1. Extracção de dados
2. Visualização de jogos
3. Detecção de eventos
4. Ontologias específicas para o futebol

2.1 Extracção de Dados

O projecto, visto ter como objectivo a análise numérica de jogos de futebol reais, tem na sua raíz duas etapas: recolha dados e análise de dados. Dada a natureza do projecto, o primeiro passo consistia na agregação de dados, de jogos de futebol, para posterior análise. As hipóteses consideradas foram as seguintes:

- Extracção de dados através da monitorização de jogos.
- Utilização de dados já extraídos por outras entidades.

Para a realização desta primeira etapa, extracção de dados reais, tornava-se necessário implementar ou utilizar um sistema de extracção de dados em jogos de futebol já existente. O sistema em causa teria de ser capaz de identificar, em tempo real, a posição dos vários agentes (jogadores) bem como a posição da bola no decorrer do jogo.Neste sentido, para que isto fosse possível, era necessário que o sistema de tracking/seguimento de objectos a utilizar, capturasse a posição de todos os intervenientes do jogo, jogadores e bola, em cada momento através de tracking individual de objectos.

Devido às restricções, como, por exemplo, a garantia de elevada precisão, dois tipos de soluções foram consideradas para o tracking de objectos:

1. Utilização de RFIDs [RZ07].
2. Utilização de RFIDs juntamente com sensores visuais (para garantir maior precisão) [JKSS07]

No entanto e devido, fundamentalmente, a questões de ordem temporal, burocrática, técnica e financeira optou-se por utilizadar dados já previamente colectados por outras entidades.

Para o desenvolvimento do projecto foram contactadas várias entidades como a FIFA [Cora] e a empresa Deltatre [Del]. A Deltrate é uma empresa que extrai informação manualmente de jogos desportivos. A empresa tem um conjunto de pessoas que monitorizam determinadas competições desportivas, comercializando os dados por si extraídos. Ambas as empresas foram contactadas com o intuito de obter dados mesmo que de competições de futebol de 11 jogadores mesmo que fossem competições de selecções nacionais dos escalões jovens. Porém essas entidades não se mostraram disponíveis para facultar os dados, mesmo tendo conhecimento que o projecto era exclusivamente académico. Nesse sentido, a solução encontrada passou pela utilização de dados referentes a jogos de futebol Robótico, provenientes da competição Robocup [Wikc] na liga de simulação.

O Robocup é uma competição internacional entre robôs(reais ou simulados) em jogos de futebol, cujo principal objectivo é promover e desenvolver a inteligência artificial [Wikc]. As semelhanças entre os jogos simulados no robocup e os jogos humanos, juntamente com a disponibilidade dos dados, tornaram o Robocup como uma excelente alternativa à falta de dados reais.

2.2 Visualização de Dados

Após a escolha do Robocup como fonte de dados a utilizar, tornou-se prático considerar os visualizadores de jogos do Robocup como visualizadores de dados. Os visualizadores considerados foram os seguintes:

- Soccer Monitor [Feda] (primeiro visualizador desenvolvido, corre na plataforma Linux).
- FrameView [Fra] (visualizador 2D oficial com várias funcionalidades de análise [Fra]).
- TeamDesigner [TS03] (visualizador 2D desenvolvido na Universidade do Porto [TS03]).
- SoccerScope [MNT02], [MNT02] (visualizador 2D, desenvolvido em Java, open-source).
- Magic Box [XLL02] (Visualizador 3D desenvolvido utilizando OpenGL).

- Visual3D [Lou04] (Visualizador 3D desenvolvido na Universidade do Porto [Lou04]).

Os requisitos para o visualizador de dados eram os seguintes:

- Rapidez.
- Funcionalidades.
- Integração com outro software.

A rapidez de execução era um factor muito importante pois pretendia-se que o software executasse numa máquina vulgar sem capacidade de processamento especial. Daí a exclusão dos visualizadores: Visual3D [Lou04] e Magic Box [XLL02], e a opção por visualizadores 2D. O protótipo que se pretendia desenvolver, iria detectar eventos num jogo de futebol robótico, se o visulizador englobasse, também ele, funções de análise de jogo, tornaria o teste do protótipo mais fácil, pois seria mais um factor de comparação de resultados, por essa razão a utilização do Soccer Monitor não foi considerada. Por fim, a integração do visualizador com o protótipo, era uma característica muito importante, era necessário uma integração entre as duas interfaces, do protótipo e do visualizador, que fosse o mais flexível possível de forma à informação fluir de uma componente para a outra facilmente, e, ao mesmo tempo, não fosse muito dispendiosa em termos temporais, de forma a que todos os objectivos fossem atingidos. Entre o FrameView [Fra], o TeamDesigner [TS03] e o SoccerScope [MNT02] a escolha foi o SoccerScope devido à enorme facilidade em extendê-lo com novas funcionalidades [Lou04].

2.3 Detecção de Eventos

Existem actualmente inúmeros métodos para a detecção de eventos, em jogos de futebol, da mais variada natureza.

2.3.1 Detecção de eventos através de análise de vídeo (imagem)

Este tipo de abordagem como descrito em [KLT03] apenas consegue detectar eventos de "alto nível", isto é, apenas é possível identificar eventos como cantos, golos ou expulsões de jogadores [KLT03]. A abordagem consiste na análise de imagem juntamente com Hidden Markov Models [JMA+02], a ideia base é que um evento encontra-se presente num conjunto consecutivo de imagens. A cada imagem é atribuído um valor mediante o que se encontra nela representado [KLT03], por exemplo, no caso de apenas estar presente o público atribui-se um valor, no caso de apenas estar presente uma baliza atribui-se outro valor. Um evento é definido como um conjunto consecutivo de valores que têm de estar presente em cada segmento consecutivo [KLT03].

2.3.1.1 Produtos

Existem, neste momento, no mercado um conjunto de produtos que utilizam algoritmos de análise de imagem para extrair informação sobre jogos de futebol. Os mais conhecidos são o prozone [Corb], utilizado por diversas equipas da "Barclays Premiere League"[Corb], e Match Analysis [Inc] que é utilizado pela selecção Alemã de futebol [Inc]. A análise de vídeo é apenas uma das suas características, são sistemas de gestão de jogadores e de equipas. Os eventos detectados, devido às limitações desta abordagem, são os descritos na secção anterior: golos, cantos e expulsões de jogadores.

2.3.2 Detecção de eventos através de análise de vídeo (áudio)

A detecção de eventos através da análise do áudio [CSZ+03] tem a mesma limitação que a abordagem anterior: só consegue detectar eventos de "alto nível"[CSZ+03]. O algoritmo é em tudo idêntico ao descrito na secção anterior: começa por dividir o video em segmento e analisa o áudio em cada um deles, mediante várias características do áudio, como por exemplo, o tom de voz do comentador ou o bararulho feito pelo público, atribui ao segmento de vídeo um determinado valor. Os eventos são definidos por uma gramática que identifica que valores, e a sua sequência, devem estar associados a cada segmento do vídeo para o evento em causa ser detectado.

2.3.3 Detecção de eventos através do tracking dos elementos do jogo

Actualmente existe no mercado uma solução integrada de recolha de dados e sua análise para extracção de eventos em jogos de futebol: Ascensio System [Sys]. Este sistema utiliza um conjunto de quatro câmeras, que têm obviamente de ser colocadas num estádio, que através de análise de imagem e triangulação são capazes de fazer o tracking tanto dos jogadores envolvidos no jogo como, também, da bola. Este sistema consegue detectar um conjunto bastante elevado de eventos: remates, passes, fora-de-jogo, offsides. No entanto, segundo a própria empresa, o sistema não é completamente autónomo, devendo os cálculos do sistema ser avaliados por uma pessoa [Sys] .

2.4 Ontologias específicas para o futebol

Uma ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferência sobre os objectos do domínio, sendo utiliza como uma forma de representação de conhecimento [Wikd], [Gua98]. As ontologias não se encontram apenas presentes no mundo académico tendo um posição bastante relevante em várias aplicações: e-commerce (Amazon e Ebay), procura na web (Yahoo e Lycos), entre outras [MD03]. Actualmente as

ontologias que mais sobresaem, na área do futebol, são: Ranwez Soccer Ontology [Ran] , que nasceu do trabalho de pesquisa de Ranwez, e a ontologia SWAN Soccer Ontology, integrada no projecto SWAN - Semantic Web Annotator [PKK+03]. A ontologia proposta por Ranwez consiste na construcção de uma narrativa utilizando determinados conceitos previamente definidos [Ran], como por exemplo: a construcção da definição de canto através dos conceitos previamente definidos: "Bola Fora", "Passe", "Stop"entre outros [Ran].

No caso do Robocup existe um ontologia oficiak, CLANG [Rei03]l: uma linguagem que define uma gramática com os conceitos que permitem aos treinadores enviarem mensagens para um determinado conjunto de jogadores, para eles efectuarem determinadas acções em certas partes do terreno de jogo. Recentemente foram adicionadas características da linguagem COACH UNILANG [Rei03] que a tornaram ainda mais expressiva.

Capítulo 3

Tecnologias

3.1 Software Utilizado

Tal como foi referido anteriormente, o objectivo principal do projecto consistia na extracção automática de eventos. Atendendo ao facto da natureza dos dados a utilizar (ficheiros de log provenientes do futebol robótico) era útil reutilizar componentes relacionadas com o Robocup. Devido à escolha previamente referida no capítulo anterior, do uso de ficheiros de log provenientes de competições do RoboCup realizadas, seria pertinente utilizar uma ferramenta já existente que permitisse ler vários formatos de ficheiros de log. Neste sentido, o foi escolhido o software SoccerScope2 [Nak] por diversas razões:

1. Linguagem de Programação Java
2. Código estruturado e documentado
3. Compatível com diferentes versões do Robocup

O facto de ter sido desenvolvido em Java garantia a presença de dois importantes factores, vitais, para o desenvolvimento deste projecto:

1. Portabilidade do código garantida entre máquinas diferentes com sistemas operativos diferentes
2. Executar na Java Virtual Machine

O facto do software em causa correr na JVM (Java Virtual Machine), permitia explorar as vantagens desta máquina virtual como por exemplo:

1. Conjunto de linguagens dinâmicas disponíveis que fazem a ligação com o código Java de forma transparente
2. Conjunto de bibliotecas disponíveis, consolidadas e devidamente testadas
3. Performance State-of-the-art [Mica], [Jac]

3.2 Tecnologias para o desenvolvimento do sistema de detecção automática de eventos

Com a escolha do software SoccerScope para auxiliar o desenvolvimento do projecto, surgiram várias linguagens de programação nas quais o projecto podia ser desenvolvido, das quais se destacam:

1. Java
2. Clojure
3. Groovy
4. JRuby (Ruby)
5. Jython (Phyton)
6. Rhino (Javascript)
7. Scala

Devido à natureza do projecto a característica mais importante era a flexibilidade da linguagem (mais que velocidade de execução). A natureza ambígua dos eventos, como o passe ou o remate, obrigava à flexibilidade do protótipo, de forma a que utilizadores avançados podessem facilmente implementar as suas próprias ontologias.

Devido a este facto a escolha teria que ser uma linguagem dinâmica orientada a objectos como JRuby, Jython ou Rhino Javascript. A escolha recaiu sobre a linguagem Ruby por duas razões: devido à familiarização do estudante com ela, e, devido à facilidade com que se constroem DSLs.

Uma DSL é uma "Domain-Specific Language", isto é, uma linguagem formal orientada a resolver um problema específico [Fre][Wike]. Neste projecto, optou-se por um desenvolvimento "bottom-up"para construir uma linguagem que permitisse a construcção de ontologias que fossem flexíveis e que fossem facéis de extender.

3.2.1 Graphical User Interface e gráficos estatísticos

Para o desenvolvimento da interface gráfica complementar ao visualizador foi escolhido o Java Swing [Micb] e o Cheri. O Swing é uma API para o desenvolvimento de GUIs em Java que tem como vantagens a sua portabilidade, o facto de utilizador obter o "look and feel"nativo (isto é, a aplicação utiliza o "aspecto"do sistema operativo em que corre) e de seguir o padrão MVC que ajuda a simplificar a lógica das aplicações.

O Cheri é uma framework para o desenvolvimento de aplicações Swing utilizando o JRuby, tem como grande vantagem o facto de ser utilizado como uma DSL, o que simplifica bastante o desenvolvimento de GUIs.

Era também necessário a representação gráfica de eventos num determinado jogo, para tal foi utilizada a biblioteca JFreeChart [Ref]. O JFreeChart é uma biblioteca, open-source, que permite o desenho de qualquer tipo de gráficos, sejam eles gráficos de barras, circulares, etc, em componentes Swing. O uso desta biblioteca, que é utilizada por grandes empresas [Ref], garante o desenvolvimento rápido e sólido das funcionalidades gráficas necessárias.

Capítulo 4

Overview do Robocup

4.1 História

A ideia do Futebol Robótico surgiu na Universidade de British Columbia, Canadá em 1992 por Alan Mackworth [Rei03]. No entanto, foi em Outubro de 1992 que um grupo de investigadores se reuniu no Japão e definiu os objectivos, as normas, e os standards pelos quais o Robocup se rege.

Foi em Setembro de 1993 que a primeira versão do simulador de futebol virtual foi apresentado ao público [Rei03]. Investigadores no Japão desenvolveram a primeira versão do simulador oficial de futebol, soccerserver. Após este lançamento, qualquer investigador podia desenvolver a sua equipa de futebol e testá-la contra outras equipas, bastando que ambas reconhecessem o protocolo de comunicação do Robocup [Rei03].

A primeira competição do Robocup aconteceu em de 1997 em Nagoya, Japão, com a presença de cerca 40 equipas num dos maiores eventos científicos de sempre [Fedb]. As edições seguintes realizaram-se em Paris (1998), Estocolmo (1999), Merlbourne(2000), Seattle (2001), Fukuoka (2002) , Padova (2003) , Lisboa, (2004), Osaka (2005), Bremen (2006), Atlanta (2007), Suzhou (2008) e Graz (2009). [Fedc]

4.1.1 Associação Reguladora

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 [Rei03].

Esta organização é a responsável pela organização do campeonato do mundo do Robocup todos os anos e contribui para o suporte à investigação nesta área [Rei03]. Existem, ainda, comités locais para auxiliar os investigadores.

4.1.2 Vertentes

O Robocup engloba várias categorias diferentes: Robocup Rescue, Robocup júnior, Robocup Soccer. [Wikc] O Robocup rescue consiste na aplicação da investigação do futebol Robótico ao domínio da busca e salvamento em caso de catástrofes. Por sua vez, o Robocup Júnior consiste numa versão do Robocup Soccer orientada a crianças, onde os intervenientes criam equipas de robôs para jogar futebol, dançar ou resgatar vítimas [Rei03]. o Robocup Soccer, sendo a origem do próprio robocup, consiste na competição de equipas de futebol numa das seguintes vertentes: [Wikc]

- Liga de Simulação - Nesta competição não existem robôs, existindo um servidor central (soccerserver) onde os clientes se ligam via rede, e enviam as suas ordens para os jogadores (agentes) virtuais [Wikf]
- Liga de Robôs Pequenos - equipas de 5 competem entre si, onde os robôs são desenhados por cada equipa sendo obrigados a seguir certas regras [Wikg]
- Liga de Robôs Médios - equipas de 6 competem entre si, onde os robôs são desenhados por cada equipa sendo obrigados a seguir certas regras [Wikh], onde os robôs comunicam via wireless.
- Liga Standard - equipas competem com robôs idênticos e totalmente autónomos, o robô Sony Aibo foi utilizado até 2007 [Wiki], sendo actualmente utilizado o robô Nao [Wikj].

4.2 Liga de Simulação

A liga de simulação foi a utilizada no projecto, pois é que apresenta os resultados mais realistas. As características dos jogadores são definidas formalmente não existindo as limitações físicas do futebol robótico. Esta liga é baseada no sistema de simulação, de utilização livre, soccerserver [Rei03]. Este sistema simula um campo contendo duas 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 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 [Rei03]. É vista como a competição ideal para ferramentas de debug e análise de agentes.

A evolução na investigação proveniente da liga de simulação tem sido bastante significativa, aproximando-a do futebol real, através da introdução de conceitos do futebol real, como: técnicas de coordenação como formações e papéis, troca de papéis e posicionamentos de agentes heterogéneos [Rei03], o conceito de táctica [Rei03] e criaram um

mecanismo de posicionamento que permite a definição de formações muito semelhantes às utilizadas no futebol real [Rei03].

4.2.1 Sistemas Multi-Agente

Os sistemas multi-agente, como é o caso da Liga de Simulação, consistem na construção de sistemas autónomos e cooperativos, com vista à resolução de um problema global [dA08]. No caso da liga de simulação os sistemas são consituídos por equipas e o objectivo comum é ganhar o jogo em disputa. Os agentes, jogadores, são caracterizados por possuirem capacidade de decisão autónoma, são capazes de comportamento reactivo, pró-activo, social e possuem um fluxo de controlo próprio distinto dos restantes agentes que compõem um dado sistema multi-agente (equipa) [dA08]. A grande vantagem deste paradigma consiste em atingir objectivos, através da cooperação e do trabalho em equipa, em que um agente por si só não seria capaz de atingir. Estes agentes exibem duas características fundamentais: serem capazes de agir de forma autónoma tomando decisões que conduzam à satisfação dos seus objectivos; serem capazes de interagir com outros agentes utilizando protocolos de interacção social inspirados nos humanos e incluindo pelo menos algumas das seguintes funcionalidades: coordenação, cooperação, competição e negociação [Rei03].

A interacção é fundamental numa sociedade de agentes, para isso é necessário saber coordená-los uns com os outros. Os sistema multi-agentes computacionais contêm agentes que interagem uns com os outros, sejam agentes homogéneos ou agentes heterogéneos, que permite que seja possível poder transportar para um ambiente de simulação aquilo que se passa no mundo real [dA08]

4.2.2 Arquitectura da Liga de Simulação

A estrutura da liga de simulação consiste numa arquitectura Cliente - Servidor. O simulador recebe os comandos dos agentes, executa-os simulando o movimento de todos os objectos no campo enviando aos agentes a informação sensorial [dA08], sendo a comunicação efectuada, como foi referido anteriormente, por sockets UDP.

4.2.2.1 Protocolos

Existe vários tipos de procolos utilizados entre os clientes e o servidor, sendo classificados da seguinte forma:

1. Protocolo de Conexão - permitem aos clientes ligarem-se e desligarem-se do servidor

2. Protocolo de Acção - definem as ordens que cada cliente envia para os seus agente, assim como, a sintaxe das respostas, possíveis, do servidor

3. Protocolo de Percepção - definem as mensagens que o servidor pode enviar ao clientes que caracterizam o estado do jogo. Estas mensagem, que são relativas a cada agente , podem retratar três tipos de informação de percepção: Visual, Auditiva e física (o que cada agente "sente"). Este tipo de mensagens são de extrema importância, as ordens enviadas para cada agente dependem do estado, da percepção, de cada um deles.

4.2.3 Agente Heterogéneos

Os jogadores da liga de simulação, ao contrário do que o que acontece na liga standard, são os agentes heterogéneos, isto é, não têm características iguais. O que na realidade acontece é o seguinte: existe um conjunto de características presentes em todos os jogadores, sendo cada característica, como por exemplo: velocidade ou a potência do remate, definida numa cerca escala, sendo a cada jogador associado o conceito de tradeoff [dA08], isto é, por cada característica que seja uma mais valia, existe outra que é uma menos valia. No ínicio do jogo a cada treinador é fornecido um conjunto de jogadores heterogéneos e é com esse grupo que ele forma a equipa [dA08]. As características de cada jogador são as seguintes:

- Velocidade máxima
- Vigor (stamina)
- Capacidade de manter a velocidade máxima (decréscimo de velocidade)
- Inércia
- Acelaração
- Tamanho
- Capacidade de Chuto
- Precisão do chuto
- Vigor Extra (Jogador com energia extra)
- Jogador com capacidade de esforço
- Jogador sem capacidade de esforço [dA08]

4.2.4 Treinador

O treinador tem como função gerir a equipa de futebol de forma a conseguir a vitória. É responsável por escolher a melhor táctica e por efectuar as substituições necessárias ao

melhor rendimento da sua equipa. Numa 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 equipa. [dA08]

Existem 2 tipos de treinadores:

1. Treinador off-line - apenas pode ser utilizado durante o desenvolvimento, e não em competições, podendo controlar o jogo, desactivar o árbitro virtual se for preciso, movimentar os jogadores da sua equipa e da equipa adversária e colocar a bola em qualquer parte do campo, bem como definir as características dos jogadores [dA08].

2. Treinador on-line - utilizado nas competições não possui as características do treinador off-line, mas pode comunicar com os jogadores quando o jogo está parado ou no decurso do jogo, através do envio de mensagens, que podem conter atrasos. O treinador recebe informação global sem erros e como não tem solicitações em tempo real, pode gastar o seu tempo efectuando análises mais complexas acerca do comportamento do adversário e da estratégia, tácticas e formações a utilizar pela sua equipa [dA08].

A linguagem normalmente utilizada pelos treinadores para comunicar com os agentes, jogadores, é a linguagem CLang [Won]. A linguagem define uma gramática com os conceitos que permitem aos treinadores enviarem a mensagens para um determinado conjunto de jogadores, para eles efectuarem determinadas acções em certas partes do terreno de jogo. Recentemente foram adicionadas características da linguagem COACH UNILANG [Rei03] que a tornaram mais expressiva.

Capítulo 5

Implementação

5.1 Interligação JRuby e Java

Utilizando o software SoccerScope, estando este escrito em Java, como foi referido anteriormente, decidiu-se utilizar a linguagem de programação Ruby para o desenvolvimento do protótipo. Foi utilizado o JRuby para fazer a ponte entre as duas linguagens. O JRuby é a implementação, em Java, da linguagem de programação Ruby [Micc], todo o código do interpretador é implementado na linguagem java e compatível com a versão 1.8 da linguagem ruby.

A grande vantagem da utilização do JRuby reside no facto da comunicação entre as linguagens ser bi-direcional, ou seja, é possível interagir com classes Java em Ruby e vice-versa. Outra vantagem importante reside no facto do código poder ser interpretado normalmente, ou, compilado para Java Bytecode, em runtime ou "ahead-of-time", tornando-o na implementação Ruby mais rápida. [Can]

5.2 Arquitectura

5.2.1 Overview

De seguida é apresentado um diagrama de classes que demonstra a arquitectura do sistema:

A class Analyser é a responsável por despoletar todo o processo de comunicação entre as linguagens, bem como, a criação de todos os objectos gráficos necessários. Esta classe encarrega-se de invocar o sistema SoccerScope passando como parâmetro um ficheiro com dados de um jogo do robocup. Todo o processamento de carregamento de dados é efectuado pelo código Java, a biblioteca JRuby encarrega-se, após o carregamento dos dados, de converter os objectos retornados pelo soccerScope, objectos Java, em objectos Ruby. Estes objectos devolvidos pelo SoccerScope consistem numa lista, em que a cada elemento da lista corresponde um determinado instante do jogo em causa, e outra lista

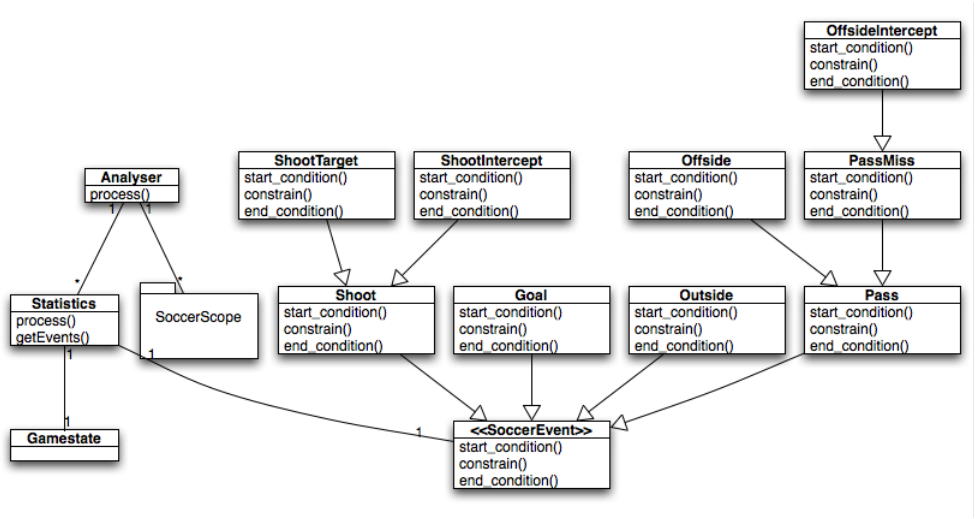

Figura 5.1: Diagrama de classes

associada com a posição de cada jogador e da bola. O SoccerScope, além do carregamento de ficheiros Robocup, é também responsável por um conjunto de operações relacionadas com a interface gráfica:

- O sistema de visualização do jogo
- O sistema de visualização de regiões dominantes por parte de uma equipa
- O sistema de visualização de passes e de modelo de forças

É precisamente no último ponto em que é necessário que o protótipo forneça ao SoccerScope a informação sobre os passes efectuados por cada jogador, de maneira ao software efectuar o desenho correctamente.

Após receber os dados do SoccerScope, estes são enviados para um objecto da classe Statistics, classe que funciona como motor do sistema.

5.2.2 Classe Statistics

A classe statistics é o grande motor do sistema. Esta classe é responsável pelo armazenamento dos dados vindos do robocup e pelo seu processamento de forma a detectar automáticamente todos os eventos.

A classe, internamente, inclui uma lista de "scenes"(cada "scene"corresponde a um dado instante do jogo e contém a informação das posições: dos jogadores e da bola), e um objecto Gamestate. O Gamestate é uma estrutura de dados, uma lista, em que era necessário uma flexibilidade muito grande. O Gamestate serve como, uma espécie de repositório de informação auxiliar ao cálculo de eventos, por outras palavras, o gamestate é uma lista, em que uma posição encontra-se indexada a um instante de tempo, onde

pode ser guardada informação, por parte dos cálculos de eventos, que sejam útil para outros eventos. Por exemplo, a identificação de um "kick", ou seja, o movimento de acelaração por parte da bola, para tornar essa informação acessível às classes responsáveis pelo cálculo dos eventos, fazia sentido ter uma estrutura de dados que informasse essas classes que uma determinada situação ocorreu. No entanto, ao protótipar a classe surge um problema: qual a informação que pretendemos transmitir ? Apenas as acelarações por parte da bola? E a acelaração dos jogadores ? E a informação de quem provocou uma acelaração na bola? Devido à extensa lista de possibilidades tornou-se necessário desenvolver uma estrutura de dados que fosse flexível ao ponto de, além de fornecer informação, fornecer também que tipo de informação era específicada. Devido a esta facto a estrutura de dados segue a "design pattern"[Wikk] "Prototype"[Wikl].

5.2.2.1 Design Pattern Prototype

Este padrão, também conhecido como padrão universal [Wikl], segue o seguinte princípio: "A ideia de que existe generalização num evento específico é de extrema importância"[Hof00] Este princípio remete-nos para a necessidade de implementar uma estrutura de dados, em que o tipo de informação a armazenar não se encontre pré-definido, podendo, além de adicionar nova informação, ser adicionado novas classes de informação. A solução consiste em implementar uma "Property List"[Yeg], esta lista é utilizada como um meio de adicionar um conjunto de atributos a um objecto em "run-time". A cada atributo é associado um nome específico que o identifica , podendo ser removido ou adicionado sem restricções.

Como exemplo podemos imaginar o seguinte problema: num determinado instante de tempo, ocorreu uma acelaração na bola (definido como chuto). Se o instante de tempo for modelado como uma "property list"podemos adicionar o valor da acelaração com o nome "kick". Mais tarde, se pretendessemos adicionar o jogador responsável pelo chuto, bastaria adicionar um objecto jogador com o nome "from".

5.2.3 Detecção de eventos

O método mais importante da classe Statistics é o método "process". Este método é responsável pela detecção de eventos num jogo do robocup. Depois das classes que definem cada eventos serem identificadas, e a lista com a informação do jogo (posições dos jogadores e da bola) devidamente preenchida, o método process itera sobre essa lista na procura de eventos.

Cada evento é modelado por uma classe com três métodos:

- start_condition
- constrain
- final_condition

Um evento é detectado caso: exista um instante de tempo (t1) em que start_condition é verdadeira, exista um instance t2 (t2 > t1) em que final_condition é verdadeira, e em todos os instantes t (t1 < t < t2) entre t1 e t2 constrain devolve o valor verdadeiro. O algoritmo pode ser descrito da seguinte maneira:

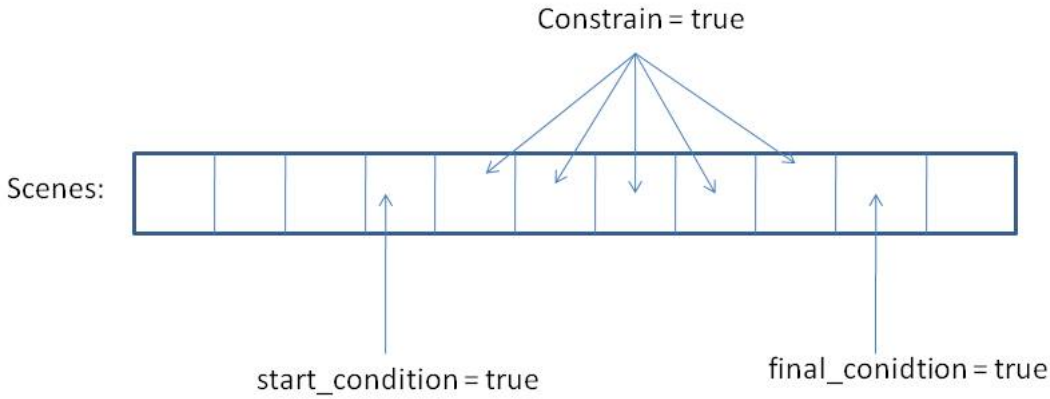

Figura 5.2: Algoritmo Genérico para detecção de eventos

O método process efectua um primeiro varrimento, onde são identificados os instante do jogo em que um determinado evento pode ter ocorrido (start_condition retornou o valor "verdadeiro"). Num segundo varrimento, para cada possível evento, é testado se os instantes de jogo seguintes seguem as restricções do evento em causa, em caso afirmativo um evento foi detectado. 1

Algorithm 1 Algoritmo para Detecção de Eventos (genérico)

```
for classe de evento do
  for i = 0 to maxcycles do
    if evento.start_condition(scene[i]) == true then
      for j = i + 1 to maxcycles do
        if evento.constrain(scene[j]) == false then
          break
        end if
      end for
      if evento.final_condition(scene[j]) == true then
        return true
      end if
    end if
  end for
end for
```

5.2.4 Classe EventInfo

Esta classe representa um evento detectado, fazendo a ponte entre o código low-level e a interface, basicamente contém a seguinte informação:

- Classe do evento
- Mensagem a aparecer na interface
- Equipa responsável pelo evento
- Número do jogador responsável pelo evento
- Instante do jogo em que o evento ocorreu (teve início)

5.3 Ontologias e Eventos

5.3.1 Estrutura

É importante realçar que para determinar qualquer evento, houve uma condição que foi assumida desde o ínicio: a velocidade da bola só pode sofrer acelaração motivada pela acção de um jogador. Isto é, assumiu-se que sempre que a velocidade da bola aumenta deve-se à acção de um jogador e não a factor externo. A acção, por exemplo, do vento sobre a bola foi descartada.

No caso do Robocup, na liga de simulação, esta premissa não levanta problema algum, visto não existir nenhuma condição externa que provoque uma acelaração positiva na velocidade da bola, no entanto, em jogos de futebol reais seria necessário um estudo mais aprofundado desta situação de maneira a reduzir os casos de erro. Ao existir, num dado momento, aumento da velocidade da bola, é sempre assumido que existiu um chuto por parte de um jogador. É este princípio que serve de base para a construcção de cada detector de cada evento.

Cada classe de evento tem de herdar,directamente ou indirectamente, da classe abstracta SoccerEvent. Esta classe, tal como já foi referido, é composta pelos métodos virtuais:

- start_condition
- constrain
- final_condition

Onde cada um retorna um valor booleano, em que a start_condition tem de ser verdadeira no primeiro instante do evento, a final_condition no último instante do evento, e o método constrain tem de ser verdadeiro nos intantes do jogo entre a condição inicial e a condição final.

5.3.2 Gramática

5.3.2.1 Métodos Auxiliares

Ao construir cada detector de evento, conclui-se que existem um conjunto de métodos que são comuns a várias classes. Os métodos em questão podem ser divididos em métodos de análise matemática e métodos de lógica de jogo. Os métodos inseridos na primeira classe, métodos de análise matemática, não entram com definições do jogo em si, apenas efectuam um conjunto de cálculos que servem como base ao métodos de lógica de jogo. Os métodos de análise matemática, que se encontram definidos num módulo com o nome "SoccerMath"são os seguintes:

1. ball_traject(p1,p2,p2) - Este método recebe como parâmetro 3 pontos. Estes pontos são os 3 pontos inciais após ser aplicada uma determinada força a um objecto, o método após calcular a velocidade e a acelaração do objecto, devolve um vector com a suposta trajectória do objecto até a sua velocidade atingir o valor zero.

2. line_intersect_region?(vec_points, region)- Este método tem como parâmetros de entrada um vector com os pontos de um segmento de recta e uma região. Devolve verdadeiro ou falso caso o segmento de recta intersecte ou não a região.

3. velocity_intensity(obj,time) - Este método recebe um objecto (agente) e um instante de tempo, devolvendo a velocidade, em módulo, nesse momento do jogo do objecto em causa.

4. velocity_vector(obj,time) - Este método recebe um objecto (agente) e um instante de tempo, devolvendo a velocidade, como um vector com duas componentes, nesse momento do jogo do objecto em causa.

O conjunto de métodos que constituem os métodos de suporte à lógica do jogo, presente no módulo "SoccerUtils", são:

1. preview(iscene, fscene) - Este método recebe como parâmetros de entrada dois instantes de tempo em que ocorreu um passe falhado, este método encarrega-se de calcular a possível trajectória da bola (caso o jogador da equipa adversária não a tivesse interceptado) devolvendo o jogador que, com a maior probabilidade, a iria receber.

   A forma como o jogador de destino é calculado é a seguinte: para cada jogador é calculada a sua distância à trajectória da bola, mediante a distância a que ele se encontra, a velocidade da bola no ponto de intercepção, e a velocidade do jogador, é seleccionado o que teria melhores condições de a receber.

2. is_kicking(scene1,scene2) - Este método indica se ocorreu um chuto (aumento da velocidade da bola) entre os momentos de jogo scene1 e scene2. Em caso positivo o jogador que chutou a bola é devolvido caso contrário é devolvido o valor falso.

3. inside_field? e outside_field? - Este métodos, tal como os nomes indicam, dizem-nos se um determinado objecto se encontra fora ou dentro de campo num determinado momento do jogo.

4. get_first_moment_wball(player,scene) - Este método recebe como parâmetros de entrada um jogador e um dado momento de jogo. Devolve o instante de tempo em que esse jogador teve a posse de bola, anteriormente ao momento scene.

5. get_attack_team(scene) - Este método indica-nos qual a equipa que no instante "scene"se encontra na posição de ataque.

6. get_players_for_scene(scene) - Este método retorna as posições de todos os jogadores num dado instante do jogo.

7. closest_players - Este método retorna as posições de todos os jogadores num dado instante do jogo, ordenados pela sua distância à bola.

5.3.2.2 Detecção de Eventos

Tal como já foi referido, cada evento é definido através da extensão da classe SoccerEvent e da implementação dos respectivos métodos virtuais. A detecção de cada evento consiste nos seguintes passos:

1. Existe um determinado momento de jogo (numa determinada situação) que corresponde a (um possível) evento

2. Existe 0 ou mais condicções que se mantêm durante os momentos do jogo seguintes

3. Existe uma determinada condição final que é verificada

O conjunto de eventos a detectar foi definido de acordo com a sua importância num jogo de futebol, tentando corresponder aos eventos normalmente associados a este jogo por parte dos "media", o conjunto é, então, composto pelos seguintes eventos:

- Pass
- PassMiss
- Shoot
- ShootIntercepted

- ShootTarget
- Goal
- Outside
- Offside
- OffsideIntercept

O seguinte diagrama de classes apresenta relação entre cada um deles:

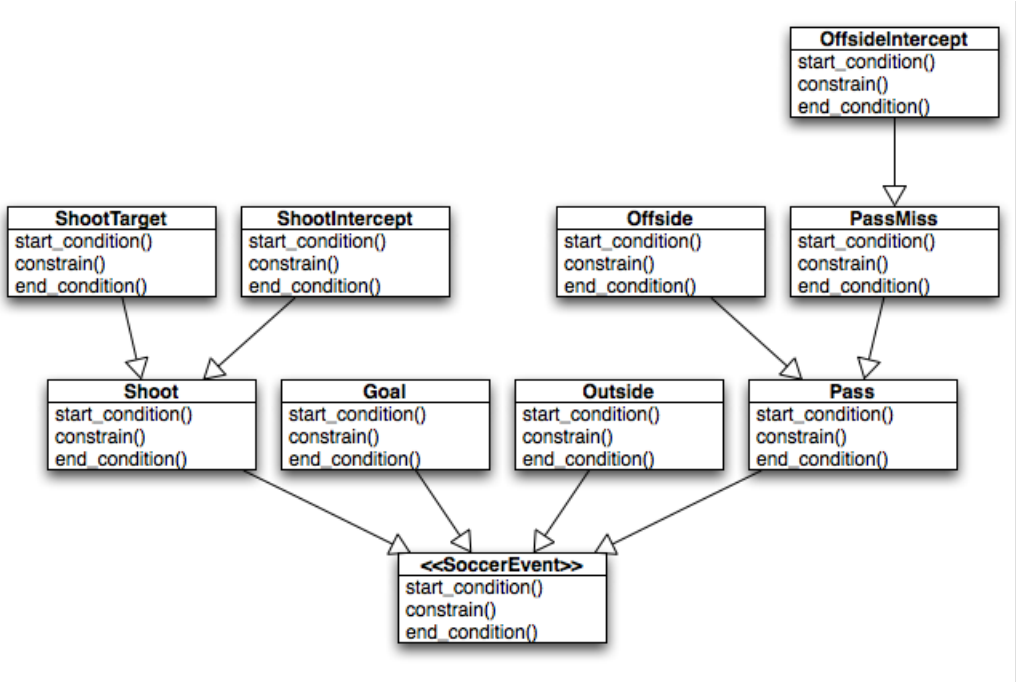

Figura 5.3: Diagrama de classes (apenas relacionadas com SoccerEvent)

5.3.2.3 Pass (extende SoccerEvent)

O evento "Pass"é definido como a troca de bola entre dois jogadores da mesma equipa. O evento é definido, formalmente, na tabela 5.1 e no algoritmo 2.

Start Condition	Constrain	Final Condition
Detecta se existe um kick, se sim guarda o jogador e a equipa responsável	Não existir nenhum kick	Existir outro kick, por parte de um jogador da mesma equipa do jogador envolvido na start condition

Tabela 5.1: Tabela Pass

Nota: É importante realçar o facto de que foi assumido que um jogador ao receber a bola irá sempre efectuar pelo menos um chuto (drible incluido).

Algorithm 2 Algoritmo para Detecção de Passes

```
for i = 0 to maxcycles do
  playerkicking ← iskicking?()
  if playerkicking = true then
    kplayer ← playerkicking[i]
    for j = i to maxcycles do
      anotherkick ← iskicking?()
      if anotherkick = true then
        break
      end if
    end for
    oplayer ← playerkicking[j]
    if kplayer.getTeam() ==oplayer.getTeam() then
      return true
    end if
  end if
end for
```

5.3.2.4 PassMiss (extende Pass)

O evento "PassMiss"é definido como a troca de bola entre dois jogadores de equipas diferentes. É também calculado o jogador para o qual o passe era, provavelmente, dirigido.

O algoritmo para identificar o jogador para o qual o passe era dirigido, baseia-se no cálculo da trajectória da bola juntamente com a velocidade e distância de cada jogador a ela. O algoritmo é exemplificado em algorithm 3 e na tabela 5.2.

Start Condition	Constrain	Final Condition
Detecta se existe um kick, se sim guarda o jogador e a equipa responsável	Não existir nenhum kick	Existir outro kick, por parte de um jogador da equipa contrária do jogador envolvido na start condition

Tabela 5.2: Tabela PassMiss

5.3.2.5 Shoot (extende SoccerEvent)

O Shoot é definido como um chuto por parte de um jogador da equipa atacante, que se encontre perto da área, em que a velocidade da bola é suficiente para atingir a linha de fundo. A tabela encontra-se definida em 5.3 e o algoritmo está formalizado em 4.

5.3.2.6 ShootIntercepted (extende Shoot)

Um ShootIntercepted é um Shoot em que a bola é interceptada por um jogador da equipa adversária, incluindo o guarda-redes. A tabela encontra-se em 5.4 e o algoritmo em 5.

Algorithm 3 Algoritmo para Detecção de Passes Falhados

```
for i = 0 to maxcycles do
    playerkicking ← iskicking?()
    if playerkicking = true then
        kplayer ← playerkicking[i]
        for j = i to maxcycles do
            anotherkick ← iskicking?()
            if anotherkick = true then
                break
            end if
        end for
        oplayer ← playerkicking[j]
        if kplayer.getTeam() ≠oplayer.getTeam() then
            return true
        end if
    end if
end for
```

Algorithm 4 Algoritmo para Detecção de Remates (Shoot)

```
for i = 0 to maxcycles do
    playerkicking ← iskicking?()
    if playerkicking.inAttackField() == true then
        kplayer ← playerkicking[i]
        for j = i to maxcycles do
            anotherkick ← iskicking?()
            if anotherkick = true then
                break
            end if
        end for
        position ← ballPosition[j]
        if position.nearGoal() == true then
            return true
        end if
    end if
end for
```

Start Condition	Constrain	Final Condition
Detecta se existe um kick, detecta se a velocidade da bola é suficiente para chegar, pelos menos à linha de fundo e se o jogador pertence á equipa atacante	Não existir nenhum kick	A bola atingir a linha de fundo,

Tabela 5.3: Tabela Shoot

5.3.2.7 ShootTarget (extende Shoot)

Um ShootTarget é um Shoot em que a bola tem a direcção da baliza da equipa adversária. A sua tabela é 5.5 e o algoritmo formalizado encontra-se em 6.

5.3.2.8 Goal (extende SoccerEvent)

Um Goal (golo) é definido como a passagem da bola de dentro do campo para dentro de uma das balizas. O algoritmo está definido em 7 e a tabela em 5.6.

5.3.2.9 Outside (extende SoccerEvent)

Um Outside é definido como a passagem da bola de dentro do campo para fora de campo. A sua tabela encontra-se em 5.7 e o seu algoritmo em 8.

5.3.2.10 Offside (extende Pass)

Para o desenvolvimento do método de detecção dos offsides foram implementados dois métodos auxiliares:

1. is_offside?(iscene,destPlayer,origPlayer)
2. is_unvalid_position(destPlayer,otherTeam)

O método is_offside? recebe como parâmetros o momento do passe, iscene, e o jogador que recebe o passe, destPlayer. Caso no momento iscene não houvesse, pelo menos 2 jogadores, cuja posição estivesse à mesma distância, ou menos, deles próprios à baliza da própria equipa, o jogador destPlayer encontra-se em posição de fora-de-jogo. A tabela encontra-se definida em 5.8 e o algoritmo está definido em 9.

5.3.2.11 OffsideIntercept (extende PassMiss)

Este evento consiste num offside não assinalado pelo árbitro. Normalmente acontece porque o árbitro deu lei da vantagem. A sua tabela encontra-se em: 5.9.

Algorithm 5 Algoritmo para Detecção de Remates Interceptados (ShootIntercepted)

```
for i = 0 to maxcycles do
  playerkicking ← iskicking?()
  if existir um jogador a chutar then
    kplayer ← playerkicking[i]
    for j = i to maxcycles do
      if existe outro chuto then
        break
      else if bola esta fora do terreno de jogo then
        balloutside ← true
      end if
    end for
    if position.nearGoal() == true AND balloutside != false then
      return true
    end if
  end if
end for
```

Algorithm 6 Algoritmo para Detecção de Remates na direcção da Baliza (ShootTarget)

```
for i = 0 to maxcycles do
  playerkicking ← iskicking?()
  if playerkicking.inAttackField() == true then
    kplayer ← playerkicking[i]
    for j = i to maxcycles do
      anotherkick ← iskicking?()
      if anotherkick == true and ballDirection == Goal then
        break
      end if
    end for
    position ← ballPosition[j]
    if position.nearGoal() == true then
      return true
    end if
  end if
end for
```

Algorithm 7 Algoritmo para Detecção de Golos

```
for i = 0 to maxcycles do
  ballPosition ← ballposition[i]
  if ballPosition.insideGoal? = true then
    return true
  end if
end for
```

Algorithm 8 Algoritmo para Detecção de Bola Fora

```
for i = 0 to maxcycles do
    ballPosition ← ballposition[i]
    if ballPosition.insideOutside? = true then
        return true
    end if
end for
```

Algorithm 9 Algoritmo para Detecção de Offsides (genérico)

```
for i = 0 to maxcycles do
    playerkicking ← iskicking?()
    if playerkicking = true then
        kplayer ← playerkicking[i]
        for j = i to maxcycles do
            anotherkick ← iskicking?()
            if anotherkick = true then
                break
            end if
        end for
        oplayer ← playerkicking[j]
        if kplayer.getTeam() == oplayer.getTeam() AND oplayer.invalidPositionIn(i) then
            return true
        end if
    end if
end for
```

Start Condition	Constrain	Final Condition
Detecta se existe um kick, detecta se a velocidade da bola é suficiente para chegar, pelos menos à linha de fundo e se o jogador pertence á equipa atacante	Não existir nenhum kick	A bola ser interceptada por um jogador da equipa adversária

Tabela 5.4: Tabela ShootIntercept

5.3.3 Divisão do campo de Jogo

De forma a facilitar o desenvolvimento de eventos que dependem da posição do campo, como por exemplo o fora-de-jogo, o campo de jogo foi divido da seguinte forma 5.4:

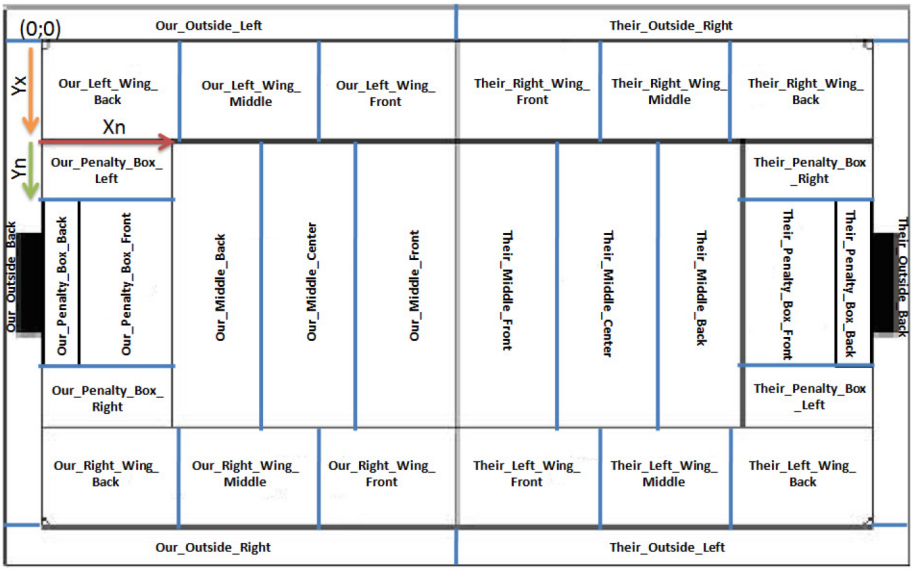

Figura 5.4: Divisão do campo

No caso do "kick", o acontecimento base para o desenvolvimento de quase todos os eventos, cada vez que é detectado, na estrutura de dados "gamestate"é, juntamente com a informação do autor do kick, guardado a região onde ocorreu o "kick". Os eventos que dependem da posição onde ocorrem são os seguintes:

- Shoot
- ShootIntercepted
- ShootTarget
- Offside
- OffsideIntercept

Start Condition	Constrain	Final Condition
Detecta se existe um kick, detecta se a velocidade da bola é suficiente para chegar, pelos menos à linha de fundo e se o jogador pertence á equipa atacante	A bola ter a direcção da	baliza da equipa adversária

Tabela 5.5: Tabela ShootTarget

Os eventos influenciados pela posição onde ocorrem são, portanto, os remates e os offsides. O caso dos remates é simples, basicamente um remate só é classificado como tal, se ocorrer no último terço do campo. Na sua definição assumiu-se que, caso ocorra um kick na direcção da baliza, por exemplo, no meio campo da equipa atacante, o que na realidade aconteceu foi um passe falhado. O caso dos offsides é ainda mais simples pois a regra do offside define que este só ocorre quando o jogador que recebe a bola encontra-se no meio campo do adversário.

5.4 GUI e gráficos

Tal como foi referido na secção "3. Tecnologias"devido ao facto do sistema a utilizar ter sido desenvolvido na linguagem Java fazia mais sentido escolher uma plataforma para o desenvolvimento da interface, que fosse ela também, escrita em Java, de forma a facilitar a integração.

A plataforma escolhida foi, como já foi referido, o Java Swing. Devido à enorme verbosidade da framework em causa, foi também utilizado o Cheri para facilitar e aumentar a rapidez do ciclo de desenvolvimento da interface. O sistema pode correr com ou sem interface gráfica, mediante as necessidades dos utilizadores. A implementação segue o modelo MVC (Model-View-Controller) estando o código relativo à detecção de eventos completamente separado do código da interface, de maneira a não sobrepôr a lógica de dois problemas distintos.

Para o desenvolvimento dos gráficos, que retratam a frequência de certos eventos, foi utilizado o jfreechart. A utilização do jfreechart revelou-se bastante simples, só sendo necessário converter os dados dos eventos no formato esperado pela biblioteca.

Start Condition	Constrain	Final Condition
A bola encontra-se dentro do terreno de jogo na sua totalidade		A bola, na sua totalidade, encontra-se dentro de uma das balizas

Tabela 5.6: Tabela Goal

Start Condition	Constrain	Final Condition
A bola encontra-se dentro do terreno de jogo na sua totalidade		A bola, na sua totalidade, encontra-se fora do terreno de jogo

Tabela 5.7: Tabela Outside

Start Condition	Constrain	Final Condition
É efectuado um passe entre jogadores da mesma equipa		O jogador que recebe a bola, no momento inicial encontrava-se em posição de fora-de-jogo

Tabela 5.8: Tabela Offside

Start Condition	Constrain	Final Condition
Um jogador tenta efectuar um passe para outro jogador da sua equipa		O jogador com maior probabilidade de receber a bola, no momento inicial, encontrava-se em posição de fora-de-jogo

Tabela 5.9: Tabela OffsideIntercept

Capítulo 6

Análise de Resultados

Para realizar a análise de resultados, foram analisados os dados obtidos através da utilização do protótipo juntamente com os dados extraídos manualmente, isto é, através da análise de uma pessoa. A análise efectuada pelo aluno não deixa de ser subjectiva, pois nos casos limite prevalece sempre a opinião pessoal, logo a classificação manual não deve ser interpretada como um método 100 % certo pois também se encontra sujeita a erro. De seguida, são analisados separadamente a detecção de cada tipo de evento. Os jogos utilizados para a realização deste estudo são do campeonato Robocup 2007 realizado em atlanta. Os jogos escolhidos foram os seguintes:

1. We 2007 0 vs 0 AT- Humboldt_Hans_Meier- a equipa We era uma das equipas mais fortes do torneio, tendo sido finalista. A equipa AT- Humboldt_Hans_Meier não tendo chegado tão longe como a We 2007 era uma equipa bastante forte.

2. Brasil2D 0 vs 18 Humboldt_Hans_Meier - O Brasil2D era uma das equipas mais fracas do torneio.

3. FCPortugal2007 3 vs 1 Nemesis - confronto entre duas equipas de competência semelhante [Tec]

4. Brainstorm 0 vs 0 AT-Humbold_Hans_Meier - jogo entre duas das equipas favoritas

5. Helios2007 0 vs 1 OPU_hana_2D - um jogo dos quartos-de-final equilibrado

6. Brainstorm 3 vs 0 We 2007 - final do torneio

Com esta escolha de jogos pretendia-se explorar uma quantidade de jogos com eventos e estatísticas muito diversas. Escolheu-se jogos entre equipas muito fortes, como a final do torneio, entre equipas fortes e equipas muito fracas, como o caso do Brasil2D, e jogos equilibrados tanto com equipas fortes como com equipas fracas. O objectivo desta escolha era testar todos os detectores de eventos com um número de casos suficientes para a validar a sua análise.

Jogo: We 2007 0-0 AT- Humboldt_Hans_Meier	Manual	Software
Nº Passes Identificados	158	153
Nº Passes correctamente identificados	158	153
Nº Jogadores bem identificados	158	152
% Passes parcialmente correctos		0.98
% Passes jogadores bem identificados		0.993
% Passes totalmente certos		0.962

Tabela 6.1: Passes correctos no jogo entre We 2007 e Humboldt_Hans_Meier

6.1 Pass

Para a análise da classificação do evento pass, foram consideradas as seguintes métricas:

- Percentagem de passes parcialmente correctos - isto é, passes que foram correctamente classificados, no entanto, o jogador de origem, ou de destino, não foi correctamente detectado
- Percentagem de passes em que o jogadores foram correctamente identificados - ou seja, do universo dos passes correctamente identificados, qual foi a pertagem desses, em que a atribuição dos jogadores, origem e destino, foi correcta
- Percentagem de Passes totalmente correcta - ou seja, quais os passes que foram devidamente identificados, incluído atribuição correcta dos jogadores: do jogador responsável pelo passe e do jogador que o recebeu.

As tabelas 6.6, 6.5, 6.4, 6.3, 6.2, 6.1 demonstram os resultados obtidos na análise do passe.

Como se pode observar pelas tabelas, a taxa de sucesso na detecção do passe situa-se entre 94.3 % e 99.2 % valores claramente satisfatórios em relação aos outros sistemas (sistemas de detecção de evento através de vídeo).

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº Passes Identificados	219	221
Nº Passes correctamente identificados	219	219
Nº Jogadores bem identificados	219	219
% Passes parcialmente correctos		0.990
% Passes jogadores bem identificados		1
% Passes totalmente certos		0.990

Tabela 6.2: Passes correctos no jogo entre Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº Passes Identificados	248	250
Nº Passes correctamente identificados	248	248
Nº Jogadores bem identificados	219	219
% Passes parcialmente correctos		0.992
% Passes jogadores bem identificados		1
% Passes totalmente certos		0.992

Tabela 6.3: Passes correctos no jogo entre FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº Passes Identificados	174	179
Nº Passes correctamente identificados	174	174
Nº Jogadores bem identificados	174	174
% Passes parcialmente correctos		0.972
% Passes jogadores bem identificados		1
% Passes totalmente certos		0.972

Tabela 6.4: Passes correctos no jogo entre BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº Passes Identificados	185	193
Nº Passes correctamente identificados	185	185
Nº Jogadores bem identificados	185	180
% Passes parcialmente correctos		0.958
% Passes jogadores bem identificados		0.972
% Passes totalmente certos		0.958

Tabela 6.5: Passes correctos no jogo entre Helius e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº Passes Identificados	252	267
Nº Passes correctamente identificados	252	252
Nº Jogadores bem identificados	252	249
% Passes parcialmente correctos		0.943
% Passes jogadores bem identificados		0.988
% Passes totalmente certos		0.943

Tabela 6.6: Passes correctos no jogo entre BrainStorm e We 2007

Jogo: We 2007 0-0 AT-Humboldt_Hans_Meier	Manual	Software
Nº Passes "Falhados"Identificados	69	80
Nº Passes "Falhados"correctamente identificados	69	69
Nº Jogadores bem identificados	69	54
% Passes "Falhados"parcialmente correctos		0.862
% Passes "Falhados"jogadores bem identificados		0.782
% Passes "Falhados"totalmente certos		0.675

Tabela 6.7: Passes falhados no jogo: We 2007 e AT-Humboldt_Hans_Meier

6.2 PassMiss

As tabelas 6.7, 6.8 , 6.9, 6.10 6.11 e 6.12 apresentam os resultado da detecção do evento "PassMiss"; ou seja, passe falhado.

A detecção dos "PassMiss", ou seja, passes falhados, apresenta resultados situados entre os 67.5 % e os 95.9 %. No entanto, se descartarmos o jogo entre We 2007 e a equipa AT- Humboldt_Hans_Meier, a taxa de detecção dos passes falhados sobre para valores entre os 92.6% e os 95.9%, o que acontece é que no jogo referido, como as equipas são de qualidade idêntica, e efectuam uma marcação forte aos jogadores da equipa adversária, existe um conjunto de situações em que é complicado definir se foi um jogador de uma equipa que recebeu a bola ou um jogador da outra equipa. Optou-se, em caso de dúvida, por considerar a classificação do protótipo como sendo errada.

6.3 Shoot

As tabelas 6.13, 6.14 , 6.15, 6.16 6.17 e 6.18 apresentam os resultado da detecção do evento Shoot (remate).

A taxa de sucesso na detecção do evento Shoot (Remate) situa-se entre 75% e os 100%. Esta grande diferença deve-se ao facto de ser muito difícil definir o que na realidade é um remate. Existem várias situações, mesmo na análise manual, em que é

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº Passes "Falhados"Identificados	118	122
Nº Passes "Falhados"correctamente identificados	118	118
Nº Jogadores bem identificados	118	113
% Passes "Falhados"parcialmente correctos		0.967
% Passes "Falhados"jogadores bem identificados		0.952
% Passes "Falhados"totalmente certos		0.926

Tabela 6.8: Passes falhados no jogo: Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº Passes "Falhados"Identificados	145	146
Nº Passes "Falhados"correctamente identificados	145	145
Nº Jogadores bem identificados	145	139
% Passes "Falhados"parcialmente correctos		0.993
% Passes "Falhados"jogadores bem identificados		0.958
% Passes "Falhados"totalmente certos		0.952

Tabela 6.9: Passes falhados no jogo: FC Portugal vs Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº Passes "Falhados"Identificados	83	85
Nº Passes "Falhados"correctamente identificados	83	83
Nº Jogadores bem identificados	83	81
% Passes "Falhados"parcialmente correctos		0.976
% Passes "Falhados"jogadores bem identificados		0.975
% Passes "Falhados"totalmente certos		0.952

Tabela 6.10: Passes falhados no jogo: BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D hline Nº Passes "Falhados"Identificados	Manual 120	Software 123
Nº Passes "Falhados"correctamente identificados	120	120
Nº Jogadores bem identificados	120	118
% Passes "Falhados"parcialmente correctos		0.975
% Passes "Falhados"jogadores bem identificados		0.983
% Passes "Falhados"totalmente certos		0.959

Tabela 6.11: Passes falhados no jogo: Helios e OPU_hana_2D

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº Passes "Falhados"Identificados	71	73
Nº Passes "Falhados"correctamente identificados	71	71
Nº Jogadores bem identificados	71	68
% Passes "Falhados"parcialmente correctos		0.972
% Passes "Falhados"jogadores bem identificados		0.957
% Passes "Falhados"totalmente certos		0.931

Tabela 6.12: Passes falhados no jogo: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº Shoots Identificados	6	7
Nº Shoots detectados correctamente	6	7
% de Shoots detectados correctamente :		0.857

Tabela 6.13: Shoot no jogo entre: We e AT- Humboldt_Hans_Meier

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº Shoots Identificados	21	25
Nº Shoots detectados correctamente	21	21
% de Shoots detectados correctamente :		0.84

Tabela 6.14: Shoot no jogo entre: Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº Shoots Identificados	6	6
Nº Shoots detectados correctamente	6	6
% de Shoots detectados correctamente :		1

Tabela 6.15: Shoot no jogo entre: FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº Shoots Identificados	1	1
Nº Shoots detectados correctamente	1	1
% de Shoots detectados correctamente :		1

Tabela 6.16: Shoot no jogo entre: BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº Shoots Identificados	3	4
Nº Shoots detectados correctamente	3	3
% de Shoots detectados correctamente :		0.75

Tabela 6.17: Shoot no jogo entre: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº Shoots Identificados	5	6
Nº Shoots detectados correctamente	5	5
% de Shoots detectados correctamente :		0.833

Tabela 6.18: Shoot no jogo entre: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº ShootsIntercepted Identificados	7	8
Nº ShootsIntercepted detectados correctamente	7	7
% de ShootsIntercepted detectados correctamente :		0.875

Tabela 6.19: Remates intercepados no jogo: We 2007 e AT- Humboldt_Hans_Meier

complicado definir se, de facto, um remate ocorreu ou não, isso resulta da ausência de uma forma directa de formalmente definir o que é um remate.

6.4 ShootIntercept

As tabelas 6.19, 6.20 , 6.21, 6.22 6.23 e 6.24 apresentam os resultado da detecção do evento ShootIntercept (remate interceptado).

A taxa de detecção de ShootIntercept, ou seja, um remate interceptado, situa-se entre os 69.9% e os 100%. Tal como o caso do Shoot, os resultados não são melhores devido à dificuldade em formalizar este evento.

6.5 ShootTarget

As tabelas 6.25, 6.26, 6.27, 6.28 6.29 e 6.30 apresentam os resultado da detecção do evento ShootTarget (remate na direcção da baliza).

Um ShootTarget (é importante realçar que os remates que resultam em golo não são considerados) é um remate na direcção da baliza. Ao contrário do Shoot a sua taxa de detecção é bastante elevada: entre 80% e 100%, devido a uma razão: ao ser especificado que o remate tem que ser na direcção da baliza, o universo de pesquisa é reduzido substancialmente, não sendo considerados os casos em que um jogador em posição de remate realiza um passe, tornando mais fácil a construcção de um algoritmo de detecção de remates na direcção da baliza.

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº ShootsIntercepted Identificados	23	33
Nº ShootsIntercepted detectados correctamente	23	23
% de ShootsIntercepted detectados correctamente :		0.696

Tabela 6.20: Remates intercepados no jogo: Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº ShootsIntercepted Identificados	7	7
Nº ShootsIntercepted detectados correctamente	7	7
% de ShootsIntercepted detectados correctamente :		1

Tabela 6.21: Remates intercepados no jogo: FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº ShootsIntercepted Identificados	4	4
Nº ShootsIntercepted detectados correctamente	4	3
% de ShootsIntercepted detectados correctamente :		0.75

Tabela 6.22: Remates intercepados no jogo: BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº ShootsIntercepted Identificados	4	5
Nº ShootsIntercepted detectados correctamente	4	4
% de ShootsIntercepted detectados correctamente :		0.8

Tabela 6.23: Remates intercepados no jogo: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº ShootsIntercepted Identificados	3	3
Nº ShootsIntercepted detectados correctamente	3	3
% de ShootsIntercepted detectados correctamente :		1

Tabela 6.24: Remates intercepados no jogo: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº ShootsTarget Identificados	7	8
Nº ShootsTarget detectados correctamente	7	7
% de ShootsTarget detectados correctamente :		0.875

Tabela 6.25: Remates à baliza no jogo: We 2007 e AT- Humboldt_Hans_Meier

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº ShootsTarget Identificados	19	19
Nº ShootsTarget detectados correctamente	19	19
% de ShootsTarget detectados correctamente :		1

Tabela 6.26: Remates à baliza no jogo: Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº ShootsTarget Identificados	0	1
Nº ShootsTarget detectados correctamente	0	0
% de ShootsTarget detectados correctamente :		1

Tabela 6.27: Remates à baliza no jogo: FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº ShootsTarget Identificados	1	1
Nº ShootsTarget detectados correctamente	1	1
% de ShootsTarget detectados correctamente :		1

Tabela 6.28: Remates à baliza no jogo: BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº ShootsTarget Identificados	4	5
Nº ShootsTarget detectados correctamente	4	4
% de ShootsTarget detectados correctamente :		0.8

Tabela 6.29: Remates à baliza no jogo: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº ShootsTarget Identificados	2	2
Nº ShootsTarget detectados correctamente	2	2
% de ShootsTarget detectados correctamente :		1

Tabela 6.30: Remates à baliza no jogo: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº Golos Identificados	0	0
Nº Golos detectados correctamente	0	0
% de Golos detectados correctamente :		1

Tabela 6.31: Golos no jogo: We 2007 e AT- Humboldt_Hans_Meier

6.6 Goal

As tabelas 6.31, 6.32, 6.33, 6.34 6.35 e 6.36 apresentam os resultado da detecção do evento Golo.

O golo é o evento mais fácil de detectar devido à facilidade com que é definido: a transposição da bola de dentro de campo para dentro de uma baliza, daí a taxa de sucesso seja 100%.

6.7 Outside

As tabelas 6.37, 6.38, 6.39, 6.40 6.41 e 6.42 apresentam os resultado da detecção do evento Outside (bola fora do terreno do jogo).

A formalização da detecção do evento Outside (bola fora do terreno de jogo), deveria ser muito similar à detecção do evento "Golo", contudo, no Robocup existem certas nuances que tiveram de ser consideradas: muitas vezes o simulador considera que a bola se encontra fora do terreno quando ela, na realidade, ainda está dentro do terreno de jogo. Esta característica quebrou um pouco a formalização do algoritmo, fazendo aparecer alguns falsos positivos, daí a taxa de sucesso se encontrar entre os 87.5% e os 100%.

6.8 Offside

As tabelas 6.43, 6.44, 6.45, 6.46 6.47 e 6.48 apresentam os resultado da detecção do evento Offside (offside este, não assinalado pelo árbitro).

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº Golos Identificados	18	18
Nº Golos detectados correctamente	18	18
% de Golos detectados correctamente :		1

Tabela 6.32: Golos no jogo: Brasil2D e Humboldt_Hans_Meier

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº Golos Identificados	4	4
Nº Golos detectados correctamente	4	4
% de Golos detectados correctamente :		4

Tabela 6.33: Golos no jogo: FC Portugal vs Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº Golos Identificados	0	0
Nº Golos detectados correctamente	0	0
% de Golos detectados correctamente :		1

Tabela 6.34: Golos no jogo: BrainStorm e AT-Humbold_Hans_Meier

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº Golos Identificados	4	5
Nº Golos detectados correctamente	4	4
% de Golos detectados correctamente :		0.8

Tabela 6.35: Golos no jogo: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº Golos Identificados	3	3
Nº Golos detectados correctamente	3	3
% de Golos detectados correctamente :		1

Tabela 6.36: Golos no jogo: brainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº Outsides Identificados	14	16
Nº Outsides detectados correctamente	14	14
% de Outsides detectados correctamente :		0.875

Tabela 6.37: Bolas Fora no jogo: We 2007 e Humboldt

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº Outsides Identificados	15	17
Nº Outsides detectados correctamente	15	17
% de Outsides detectados correctamente :		0.882

Tabela 6.38: Bolas Fora no jogo: Brasil2d e Humboldt

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº Outsides Identificados	8	8
Nº Outsides detectados correctamente	8	8
% de Outsides detectados correctamente :		1

Tabela 6.39: Bolas fora no jogo: FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº Outsides Identificados	26	26
Nº Outsides detectados correctamente	26	26
% de Outsides detectados correctamente :		1

Tabela 6.40: Bolas Fora no jogo: BrainStorm e Humbold

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº Outsides Identificados	18	19
Nº Outsides detectados correctamente	18	18
% de Outsides detectados correctamente :		0.947

Tabela 6.41: Bolas Fora no jogo: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº Outsides Identificados	20	20
Nº Outsides detectados correctamente	20	20
% de Outsides detectados correctamente :		1

Tabela 6.42: Bolas Fora no jogo: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº offsides Identificados	0	0
Nº offsides detectados correctamente	0	0
% de offsides detectados correctamente :		1

Tabela 6.43: Offsides no jogo: We 2007 e Humboldt

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº offsides Identificados	3	3
Nº offsides detectados correctamente	3	3
% de offsides detectados correctamente :		1

Tabela 6.44: Offsides no jogo: Brasil2D e Humboldt

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº offsides Identificados	1	1
Nº offsides detectados correctamente	1	1
% de offsides detectados correctamente :		1

Tabela 6.45: Offsides no jogo: FC Portugal e Nemesis

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº offsides Identificados	0	0
Nº offsides detectados correctamente	0	0
% de offsides detectados correctamente :		1

Tabela 6.46: Offsides no jogo: BrainStorm e Humbold

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº offsides Identificados	2	2
Nº offsides detectados correctamente	2	2
% de offsides detectados correctamente :		1

Tabela 6.47: Offsides no jogo: Helios e OPU

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº offsides Identificados	1	1
Nº offsides detectados correctamente	1	1
% de offsides detectados correctamente :		1

Tabela 6.48: Offsides no jogo: BrainStorm e We 2007

Jogo: We 2007 0 vs 0 AT- Humboldt_Hans_Meier	Manual	Software
Nº offsidesIntercepted Identificados	1	1
Nº offsidesIntercepted detectados correctamente	1	1
% de offsidesIntercepted detectados correctamente :		1

Tabela 6.49: OffsidesIntercepted no jogo: BrainStorm e We 2007

A taxa de detecção do Offside (neste caso o evento offside consiste num offside que não é assinalado por parte do árbitro, pois a equipa, que se encontrava a defender, recuperou a bola, o árbitro deixa por isso, o jogo prosseguir) é de 100%, visto, o offside ser evento muito fácil de formalizar: no momento do passe, se o jogador, que recebe o passe ,estiver no meio-campo de ataque da sua equipa e não estiver, pelo menos em linha com o penúltimo jogador da equipa adversária (o último também tem de se encontrar em linha, ou mais próximo da sua baliza), então ocorreu um fora de jogo.

6.9 OffsideIntercept

As tabelas 6.54, 6.54, 6.54, 6.54 6.54 e 6.54 apresentam os resultado da detecção do evento Offside (offside assinalado pelo árbitro).

A taxa de sucesso na detecção do OffsideIntercept (um fora de jogo que o árbitro assinala) situa-se entre os 85.7% e os 100% devido à facilidade de formalização do evento em causa.

Jogo: Brasil2D 0 vs 18 Humboldt_Hans_Meier	Manual	Software
Nº offsidesIntercepted Identificados	4	4
Nº offsidesIntercepted detectados correctamente	4	4
% de offsidesIntercepted detectados correctamente :		1

Tabela 6.50: OffsidesIntercepted no jogo: BrainStorm e We 2007

Jogo: FCPortugal2007 3 vs 1 Nemesis	Manual	Software
Nº offsidesIntercepted Identificados	12	14
Nº offsidesIntercepted detectados correctamente	12	12
% de offsidesIntercepted detectados correctamente :		0.857

Tabela 6.51: OffsidesIntercepted no jogo: BrainStorm e We 2007

Jogo: Brainstorm 0 vs 0 AT-Humbold_Hans_Meier	Manual	Software
Nº offsidesIntercepted Identificados	1	1
Nº offsidesIntercepted detectados correctamente	1	1
% de offsidesIntercepted detectados correctamente :		1

Tabela 6.52: OffsidesIntercepted no jogo: BrainStorm e We 2007

Jogo: Helios2007 0 vs 1 OPU_hana_2D	Manual	Software
Nº offsidesIntercepted Identificados	1	1
Nº offsidesIntercepted detectados correctamente	1	1
% de offsidesIntercepted detectados correctamente :		1

Tabela 6.53: OffsidesIntercepted no jogo: BrainStorm e We 2007

Jogo: Brainstorm 3 vs 0 We 2007	Manual	Software
Nº offsidesIntercepted Identificados	2	2
Nº offsidesIntercepted detectados correctamente	2	2
% de offsidesIntercepted detectados correctamente :		1

Tabela 6.54: OffsidesIntercepted no jogo: BrainStorm e We 2007

Capítulo 7

Conclusões e perspectivas de trabalho futuro

7.1 Conclusões

No decorrer do desenvolvimento da dissertação o Robocup demonstrou ser uma excelente plataforma para testes de algoritmos, tendo a ausência de ruído facilitado o desenvolvimento. O Robocup é uma plataforma importante que incentiva e motiva a exploração de novas área relacionadas com inteligência artificial e sistemas multi-agentes, não apenas pelas suas característica tecnológias, mas também, pela sua comunidade onde se incentiva a partilha e troca de ideias.

Observando o capítulo anterior torna-se claro que certos detectores de eventos tem uma performance melhor que outros, isto deve-se à ambiguidade na definição da cada evento. A dificuldade em formalizar cada evento levou a construcção de um protótipo que facilitasse a extensão do código e promovê-se a flexibilidade, tal verificou-se fundamental no desenvolvimento do projecto pois possibilitou o desenvolvimento ágil do projecto, com ciclos rápidos de desenvolvimento e teste. A linguagem Ruby revelou-se uma escolha acertada, não só pela sua integração com o SoccerScope, mas também por permitir um estilo de desenvolvimento menos rígido, devido às suas características dinâmicas, que permitiu que houvesse tempo para o refinamento de vários algoritmos. Pretende-se, no futuro, testar os algoritmos com dados de jogos de futebol reais, de maneira a estudar o seu comportamento perante dados com ruído.

7.2 Perspectivas de Trabalho Futuro

Associadas a este projecto existem um conjunto de possíveis tarefas a realizar num futuro próximo, podendo essas tarefas ser organizadas nas seguintes categorias:

1. Melhorias ao nível da eficiência

2. Melhorias ao nível do desenvolvimento e testes
3. Melhorias relacionadas com a detecção de eventos

7.2.1 Melhorias ao nível da Eficiência

Como seria previsível, o grande bottleneck do protótipo é o sistema de detecção de eventos. No entanto, quando o sistema foi desenvolvido esta variável foi equacionada e os detectores de eventos foram desenvolvidos de maneira a não dependerem uns dos outros. Devido a este facto, paralelizar o cálculo dos detectores dos algoritmos torna-se uma tarefa trivial, bastando, cada unidade de processamento, receber uma cópia dos dados do jogo.

7.2.2 Melhorias ao nível do desenvolvimento e testes

O ciclo de desenvolvimento de um detector de evento envolve muitos testes e constante refinamento. O processo de testes manualmente é o processo lento, monótono em que facilmente erros podem acontecer. É viável no futuro, desenvolver uma componente na interface que facilitasse o testes dos algoritmos. Para tal bastava que no primeiro teste manual, o utilizar marcasse os eventos como sendo de um determinado tipo, e mais tarde, o sistema encarregava-se de comparar os resultados dos algoritmos com os eventos marcados (apenas uma vez por jogo) pelo utilizador.

7.2.3 Melhorias relacionadas com a detecção de eventos

Seria interessante, no futuro, que existisse um refinamento dos algoritmos: poderia existir uma gramática em que se especificava que um determinado eventos não podia ocorrer em simultâneo com outro evento, por exemplo, quando um passe é efectuado mas não é completado porque o jogo terminou, não devia ser marcado como passe falhado, e se tal gramática existisse, este caso, e outros, poderiam ser tratados mais facilmente. Seria também interessante incluir algoritmos de inteligência artifical para a detecção, por exemplo, de formações. Facilmente poderia-se, também, devido à óptima taxa de sucesso da detecção de passes, desenvolver algoritmos para detectar jogadas estudadas ou trocas de bola comuns entre jogadores, de maneira a acrescentar uma mais valia ao sistema.

Referências

[Can] Antonio Cangiano. Disponível em http://antoniocangiano.com/2008/12/09/the-great-ruby-shootout-december-2008/, acedido a última vez em 28 de Junho de 2009.

[Cora] Fifa Corp. Disponível em http://www.fifa.com/en/marketing/newmedia/index/0,3509,10,00.html, acedido a última vez em 28 de Junho de 2009.

[Corb] Prozone Corp. Disponível em http://www.prozone.com, acedido a última vez em 28 de Junho de 2009.

[CSZ+03] C. Chen, Shyu, C. Zhang, L. Luo e M. Chen. Detection of soccer goal shots using joint multimedia features and classification rules. In *Proceedings of the Fourth International Workshop on Multimedia Data Mining*, 2003.

[dA08] Rui Manuel Figueiredo de Almeida. Análise e previsão das formações das equipas no domínio do futebol robótico simulado. Master's thesis, Faculdade de Economia da Universidade do Porto", 2008.

[Del] Deltatre. Disponível em http://www.deltatre.com/, acedido a última vez em 28 de Junho de 2009.

[Dun99] Eric Dunning. *The development of soccer as a world game*. Routledge, Fourth edition, 1999.

[Feda] Robocup Federation. Disponível em http://sserver.sourceforge.net/, acedido a última vez em 28 de Junho de 2009.

[Fedb] Robocup Federation. Disponível em http://www.robocup.org/games/97nagoya/311.html, acedido a última vez em 28 de Junho de 2009.

[Fedc] Robocup Federation. Disponível em http://www.robocup2009.org/114-0-welcome-to-Graz.html, acedido a última vez em 28 de Junho de 2009.

[Fra] FrameView. Disponível em http://lsl-www.cs.unidortmund.de/~merke/software/frameview/index.html, acedido a última vez em 28 de Junho de 2009.

[Fre] Jim Freeze. Disponível em http://www.oreillynet.com/ruby/blog/2005/12/what_is_a_dsl.html, acedido a última vez em 28 de Junho de 2009.

[Gua98] N Guarino. Formal ontology in information systems. In *Proceedings of the 1st International Conference une*, 1998.

[Hof00] Douglas R Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Penguin, 20th anniversary edition edition, 2000.

[Inc] Match Analysis Inc. Disponível em http://www.matchanalysis.com, acedido a última vez em 28 de Junho de 2009.

[Jac] Matthias Jacob. Cross-architectural performance portability of a java virtual machine implementation. Disponível em http://www.usenix.org/event/jvm02/full_papers/jacob/jacob_html/index.html, acedido a última vez em 28 de Junho de 2009.

[JKSS07] Lee Jinseok, Park Kyoung-Su e Hong Sangjin. Object tracking based on rfid coverage visual compensation in wireless sensor network. In *Circuits and Systems*, 2007.

[JMA+02] JAssfalg, MBertini, A.Bhbo, W.Nunziati e P. Pala. Soccer highlights detection and rewgnition using hmms. In *Prof ICME*, 2002.

[KLT03] Yu-Lin Kang, Joo-Hwee Lim e Qi Tian. Soccer video event detection with visual keywords. In *Communications and Signal Processing*, 2003.

[Lou04] Sérgio Louro. Sistema multi-agente para controlo de câmeras inteligentes. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2004.

[MD03] Mcguinness e L. D. *Ontologies Come of Age*. MIT Press, 1th edition edition, 2003.

[Mica] Sun Microsystems. Disponível em http://java.sun.com/performance/reference/whitepapers/5.0_performance.html, acedido a última vez em 28 de Junho de 2009.

[Micb] Sun Microsystems. Disponível em http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/package-frame.html, acedido a última vez em 28 de Junho de 2009.

[Micc] Sun Microsystems. Disponível em http://jruby.codehaus.org/, acedido a última vez em 28 de Junho de 2009.

[MNT02] Shinichi Miyazawa, Koji Nakayama e Ikuo Takeuchi. Soccerscope3. Technical report, Universidade Electro-Communications, Japão, 2002.

[Nak] Koji Nakayama. Disponível em http://ne.cs.uec.ac.jp/~koji/SoccerScope/index.htm, acedido a última vez em 28 de Junho de 2009.

[PKK+03] B Popov, A Kiryakov, A Kirilov, D Manov e Goranov. Semantic annotation platform. In *Intl. Semantic Web Conference*, 2003.

[Ran] Sylvie Ranwez. Disponível em http://www.daml.org/ontologies/273, acedido a última vez em 28 de Junho de 2009.

[Ref] Object Refinery. Disponível em http://www.jfree.org, acedido a última vez em 28 de Junho de 2009.

[Rei03] Luís Paulo Reis. *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2003.

[RZ07] Rhamati e Lin Zhong. Reliability techniques for rfid-based object tracking applications. In *Dependable Systems and Networks*, 2007.

[Sys] Amisco Systems. Disponível em http://www.footballsoftpro.com/, acedido a última vez em 28 de Junho de 2009.

[Tec] Georgia Tech. Disponível em http://wiki.cc.gatech.edu/robocup/index.php/Results#GroupE, acedido a última vez em 28 de Junho de 2009.

[TS03] Cláudio Teixeira e Johnny Santos. Fc portugal - uma equipa da liga de simulação do robocup. Technical report, Universidade de Aveiro, 2003.

[Wika] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Association_football, acedido a última vez em 28 de Junho de 2009.

[Wikb] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Laws_of_the_Game, acedido a última vez em 28 de Junho de 2009.

[Wikc] Wikipedia. Disponível em http://en.wikipedia.org/wiki/RoboCup, acedido a última vez em 28 de Junho de 2009.

[Wikd] Wikipedia. Disponível em http://pt.wikipedia.org/wiki/Ontologia_(ciência_da_computaç~ao), acedido a última vez em 28 de Junho de 2009.

[Wike] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Domain-specific_programming_language, acedido a última vez em 28 de Junho de 2009.

[Wikf] Wikipedia. Disponível em http://en.wikipedia.org/wiki/RoboCup_Simulation_League, acedido a última vez em 28 de Junho de 2009.

[Wikg] Wikipedia. Disponível em http://en.wikipedia.org/wiki/RoboCup_Small_Size_League, acedido a última vez em 28 de Junho de 2009.

[Wikh] Wikipedia. Disponível em http://en.wikipedia.org/wiki/RoboCup_Middle_Size_League, acedido a última vez em 28 de Junho de 2009.

[Wiki] Wikipedia. Disponível em http://en.wikipedia.org/wiki/AIBO, acedido a última vez em 28 de Junho de 2009.

[Wikj] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Nao_(robot), acedido a última vez em 28 de Junho de 2009.

[Wikk] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Design_pattern_(computer_science), acedido a última vez em 28 de Junho de 2009.

[Wikl] Wikipedia. Disponível em http://en.wikipedia.org/wiki/Prototype_pattern, acedido a última vez em 28 de Junho de 2009.

[Won] Yuk Wah Wong. Disponível em http://www.cs.utexas.edu/users/ml/wasp/robocup-clang.html, acedido a última vez em 28 de Junho de 2009.

[XLL02] Xuan Xing, Qingrui Li e Chang Liu. An intuitive 3d monitor system with automatic commentary for robocup 2002. Technical report, AI Center, University of Science and Technology of China, 2002.

[Yeg] Steve Yegge. Disponível em http://steve-yegge.blogspot.com/2008/10/universal-design-pattern.html, acedido a última vez em 28 de Junho de 2009.