Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2014

André de
Oliveira Nunes

Controlo de Bola Utilizando o Corpo do Robô CAMBADA

André de
Oliveira Nunes

Controlo de Bola Utilizando o Corpo do Robô CAMBADA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor José Nuno Panelas Nunes Lau, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Professor Doutor António José Ribeiro Neves, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Professor Doutor Artur José Carneiro Pereira** Professor Auxiliar da Universidade de Aveiro
vogais / examiners committee	**Professor Doutor Paulo José Cerqueira Gomes da Costa** Professor Auxiliar da Universidade do Porto (Arguente Principal)
	Professor Doutor José Nuno Panelas Nunes Lau Professor Auxiliar da Universidade de Aveiro (Orientador)

agradecimentos

Gostaria de agradecer aos meus pais, pela constante paciência e apoio durante esta longa viagem.
Aos meus irmãos, pelos conselhos e por serem modelos a seguir.
À Clara, pois sem ti eu não seria o mesmo. Para ti um beijo muito especial.
À família e amigos, pelo carinho e camaradagem, desde os primeiros anos.
Queria agradecer a toda a equipa do projeto CAMBADA, especialmente aos orientadores, Nuno Lau e António Neves, pela constante disponibilidade e por me ajudarem a finalizar este projeto.

Resumo

O Futebol robótico é uma excelente aplicação para promover o desenvolvimento de equipas de robôs. Nesta aplicação, a manipulação da bola é talvez a ação mais importante. Os robôs CAMBADA usam um dispositivo ativo de $grabber$ e de $kicker$ para controlar a bola, quer seja para rematar, fazer passes, receber a bola ou conduzir a bola. Por vezes pode ser vantajoso usar o corpo do robô para estes tipos de controlo de bola, quer seja por causa de avaria nos dispositivos de controlo de bola, quer seja por maior facilidade e rapidez de execução de ações.

Este trabalho apresenta alguns modelos desenvolvidos para controlar a bola com o corpo do robô CAMBADA, quer seja com a bola parada ou em movimento. Para a bola parada foram realizadas várias abordagens, com o robô a ir direto à bola ou a usar pontos intermédios para obter melhores resultados. Em bola em movimento foi desenvolvido um modelo para desviar a bola para a baliza na marcação de cantos. Foi ainda desenvolvido um modelo para um caso especial, que é a reposição da bola em jogo com o corpo do robô em situações de avaria dos dispositivos de $grabber$ ou de $kicker$.

Os resultados obtidos mostram o sucesso da maioria dos algoritmos desenvolvidos, permitindo ter 2 modelos funcionais e integrados no $software$ do projeto CAMBADA.

Abstract

The robotic soccer is an excellent program to promote the development of robot teams. In this application, the handling of the ball is perhaps the most important action. CAMBADA's robots uses a grabber and a kicker active device to control the ball, whether is to shoot to goal, make passes, get the ball or drive the ball. Sometimes it may be beneficial to use the robot's body for these types of ball control, whether is because of damage to the ball control devices, or for ease and speed the execution of actions.
This paper presents some models developed to control the ball with CAMBADA's robot body, with the ball stopped or in motion. With the ball stopped, there were developed several approaches, with the robot going straight to the ball or using intermediate points for best results. With the ball in motion, a model was developed to deflect the ball to the goal in corner kicks. It was also developed a model for a special case, which is the reposition of the ball in play with the robot's body in situations of grabber or kicker devices failure.
The results show the success of most of the developed algorithms, allowing to have two functional and integrated models in the CAMBADA's project software.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

MSL	*Middle Size League*
CAMBADA	**C***ooperative* **A***utonomous* **M***obile ro***B***ots w/* **A***dvanced* **D***istribuited* **A***rchitecture*
FIFA	*Fédération Internationale de Football Association*
PC	Computador Pessoal
ATRI	Atividade Transversal em Robótica Inteligente
IEETA	Instituto de Engenharia Electrónica e Telemática de Aveiro
RTDB	*Real Time Database*
HWcomm	*Hardware Comunication Process*
BDI	*Belief–Desire–Intention*

Capítulo 1

Introdução

Esta dissertação está inserida no projeto **C***ooperative* **A***utonomous* **M***obile ro***B***ots w/* **A***dvanced* **D***istribuited* **A***rchitecture* (CAMBADA) [1], desenvolvido na Universidade de Aveiro, pelo Departamento de Eletrónica, Telecomunicações e Informática em parceria com o Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA). CAMBADA é uma equipa de futebol robótico que participa na MSL da *RoboCup* [2]. É um projeto multidisciplinar, que envolve conhecimentos de mecânica, inteligência artificial, eletrónica de potência, instrumentação, arquitetura de computadores, telecomunicações, engenharia de software, visão e cooperação artificial, entre outros.

Os robôs CAMBADA usam desde 2013 uma nova plataforma [3], que reutilizou o modelo e funcionalidades que eram eficientes na plataforma antiga, mas tem alterações relativamente à mesma. As mais evidentes são a novo dispositivo de *grabber* e a nova forma hexagonal assimétrica, que permite explorar as laterais do robô para drible sem a necessidade de "encaixar"a bola com o sistema de *grabber*.

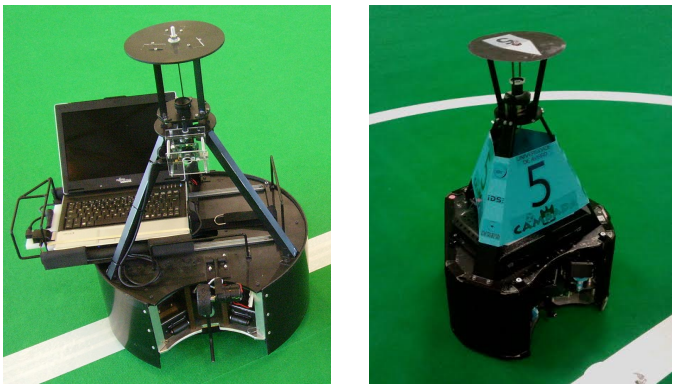

Figura 1.1: Plataforma antiga à esquerda e plataforma nova à direita.

1.1 MSL

A MSL é uma competição de futebol robótico em que cada equipa compete com um máximo de 5 robôs e uma *Base Station*, usando uma versão adaptada das regras da *Fédération*

Internationale de Football Association (FIFA). Os jogos são efetuados num campo de 18m x 12m de carpete verde com linhas brancas com uma bola oficial FIFA tamanho 5. Os robôs são completamente autónomos, têm todos os sensores e equipamento computacional incorporados no próprio robô e têm limite de tamanho e peso, tendo de caber numa caixa de 52 x 52 x 80cm e não podendo pesar mais de 40Kg.

A única intervenção humana durante o jogo vem de um árbitro, através de uma aplicação dedicada chamada *Referee Box* [4], que é controlada durante o jogo por um árbitro assistente. Ligações sem fios são permitidas somente entre robôs e entre robôs e *Base Station*, através de *unicast* ou *multicast*, não sendo permitido usar ligações *ad hoc* e *broadcast*.

Durante o jogo, de 15 minutos cada parte, a *Base Station* não pode ser mexida. Muitas equipas usam locomoção holonómica baseada numa configuração de três ou quatro rodas, assim como visão omnidirecional, por vezes em conjunto com visão panorâmica e 3D. O drible da bola pode ser feito através de dispositivos ativos ou passivos, desde que não cubra mais de um terço da bola. As equipas podem usar robôs heterogéneos, em que o guarda-redes é um caso especial, podendo usar uma estrutura extensível, em que momentaneamente (até 1 segundo) pode estender o robô para a direita, para a esquerda ou para cima, até um tamanho de 60 x 60cm de largura e até 90cm de altura [5].

Figura 1.2: Jogo da competição MSL.

1.2 Motivação

No universo MSL, e em particular nos robôs da equipa CAMBADA, é através do controlo da bola que se realizam as operações de remate, passe, receção de bola, condução de bola, etc. Estas operações são realizadas usando 2 dispositivos ativos de controlo de bola, o *grabber* e o *kicker*. O *grabber* está colocado na frente do robô e exerce sobre a bola uma força que a puxa para a zona em que a bola "encaixa" no robô e fica em posição de ser chutada ou driblada. O *kicker* permite que a bola seja chutada pelo robô para executar remates ou passes. A utilização destes dispositivos implica uma perda de tempo considerável para "encaixar" a bola na frente do robô.

O controlo de bola através da utilização do movimento e do corpo do robô permite um jogo

mais rápido, pois a bola não tem de encaixar no robô. Permite executar jogadas ao primeiro toque, tornando o controlo da bola mais próximo da forma como os humanos o fazem. Este tipo de controlo da bola permite um melhoria na robustez do sistema, sendo possível fazer passes ou remates mesmo quando os dispositivos de *grabber* ou de *kicker* estão avariados.

1.3 Objetivos

Este trabalho tem como objetivo o desenvolvimento de comportamentos de controlo de bola usando o corpo do robô CAMBADA. O robô controlará a bola chocando contra esta ou empurrando-a de modo a que o movimento resultante cumpra os objetivos desejados. Estes objetivos são dependentes do tipo de ação a executar: rematar, passar, receber a bola, conduzir a bola, etc. O desenvolvimento destes comportamentos será realizado determinando qual a trajetória que o robô deve ter para obter o efeito pretendido.

1.4 Estrutura

No Capítulo 2 apresenta-se o projeto CAMBADA e identifica-se os pontos chave tanto do software como do hardware. No Capítulo 3 apresentará o trabalho desenvolvido ao longo desta dissertação, desde o desenvolvimento dos modelos e abordagens utilizadas, assim como a implementação dos mesmo no projeto CAMBADA. No Capítulo 4 é feita uma análise dos resultados obtidos dos testes aos vários modelos. Para finalizar, no Capítulo 5 apresentam-se conclusões sobre o trabalho desenvolvido e eventuais melhorias para o futuro.

Capítulo 2

CAMBADA

O trabalho desenvolvido nesta dissertação é específico para o ambiente da MSL, mais especificamente no projeto CAMBADA.

O projeto CAMBADA começou em 2003, coordenado pelo grupo Atividade Transversal em Robótica Inteligente (ATRI) do IEETA. A equipa compete em eventos nacionais e internacionais, contando já com vários títulos de prestígio (RoboCup: 5º lugar em 2007, 1º lugar em 2008 e 3º lugar em 2009, 2010, 2011, 2013 e 2014; 2º lugar no German Open em 2010; 3º lugar no Dutch Open em 2012; 2º lugar no Iran Open 2014; Festival Nacional de Robótica: 3º lugar em 2006, 1º lugar em 2007, 2008, 2009, 2010, 2011 e 2012 e 2º lugar em 2013 e 2014). Para além das competições em campo, a equipa tem conseguido boas qualificações nos *Technical Challenges* do RoboCup (2º lugar em 2008 e 2014, 4º lugar em 2010, 1º lugar em 2009, 2012 e 2013) assim como nos *Free Challenges/Scientific Challenges* (1º lugar em 2011 e 2014 e 2º lugar em 2012 e 2013).

Este projeto inclui várias pessoas a trabalhar na estrutura mecânica dos robôs, na sua arquitetura e controladores de hardware, assim como no desenvolvimento de software em áreas como análise e processamento de imagens, fusão de informação e sensores, controlo, abordagem de deteção cooperativa baseada em base de dados de tempo-real, comunicação entre robôs e desenvolvimento da *Base Station*. A equipa CAMBADA é composta por 6 robôs, 5 deles iguais, usados como jogadores de campo e 1 guarda-redes, com algumas alterações, sendo a mais notória a inclusão de uma armação fixa.

Figura 2.1: Robôs da equipa CAMBADA.

2.1 Hardware

Os robôs foram totalmente projetados e construídos dentro do projeto CAMBADA. A base para a construção dos robôs é uma forma hexagonal assimétrica, com uma estrutura mecânica modular e por camadas, podendo substituir facilmente cada camada por uma equivalente. A camada inferior é basicamente composta por motores, rodas e baterias. A segunda camada é composta por um *kicker* eletromagnético, o sistema de *grabber* e eletrónica de controlo. A camada superior contém um Computador Pessoal (PC), um sistema de visão omnidirecional e uma bússola digital. Os robôs têm movimento holonómico devido ao uso de 3 rodas omnidirecionais, com 3 motores DC espaçados em ângulos de 120º, transmitindo força às rodas.

Os robôs seguem uma paradigma biomórfico [6], centrado numa unidade principal de processamento (o PC), que é responsável pela coordenação de comportamentos de alto nível. Esta unidade de processamento trata da comunicação com os outros robôs e tem ao seu encargo sensores com alta largura de banda, tipicamente o sistema de visão. A unidade de processamento também recebe informação de sensores com baixa largura de banda e envia comandos de atuação para controlar as atitudes do robô através de um sistema distribuído de baixo nível de sensores/atuadores (Figura 2.2).

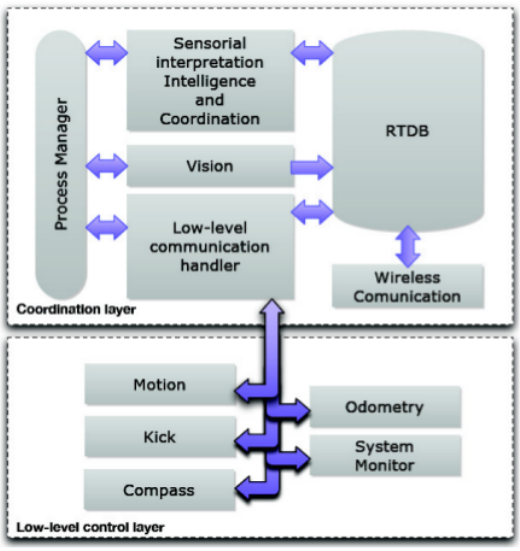

Figura 2.2: Ligação dos vários componentes de software à RTDB [7].

2.2 RTDB

No projeto CAMBADA é necessária a partilha de informação entre agentes. É com base nessa partilha de informação que a equipa decide as suas tarefas e age de forma cooperativa [8]. O objetivo da existência duma *Real Time Database* (RTDB) é a comunicação dos processos inter-robô e intra-robô e sincronização dos processos inter-robô. Apenas os processos locais podem manipular a informação que tem origem no próprio robô. A sincronização inter-robô é conseguida através de uma zona partilhada difundida pelo canal de comunicação, sendo que cada interveniente tem um *slot* seu. A zona partilhada da RTDB em cada robô é composta

pelo conjunto de todas as zonas partilhadas dos robôs. Cada agente tem um *time slot* definido para a transmissão dos seus dados em *multicast*, impedindo que o canal de transmissão seja mal utilizado. Geralmente o mecanismo de *time sharing* da unidade de processamento em sistemas operativos de uso geral não é orientado a aplicações de tempo real, o que tornaria impossível sincronizar as transmissões em intervalos de tempo bem definidos. As informações partilhadas por cada robô têm um cariz temporal, que é calculado aquando dos pedidos locais à RTDB.

A constante partilha de dados em todos os agentes permite que em qualquer momento, um agente possa tomar uma decisão comportamental baseado na informação obtida de outros agentes. Um exemplo é caso da partilha da posição da bola, podendo um agente que não tem a bola no seu ângulo de visão continuar as suas tarefas, baseado na posição da bola partilhada.

2.3 HWcomm

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

2.4 Visão

A visão é o principal subsistema de dados de entrada para o agente, recolhendo informações que permitem obter a sua localização, a localização da bola e detetar obstáculos no campo. Este subsistema baseia-se num sistema de visão omnidirecional catadióptrico [6], sendo constituído por uma câmara de vídeo digital apontada a um espelho hiperbólico. O robô com a função de guarda-redes tem uma câmara Kinect [3] para ajudar na deteção de bola num referencial 3D. A arquitetura geral do sistema de visão da equipa CAMBADA pode ser observada na Figura 2.3.

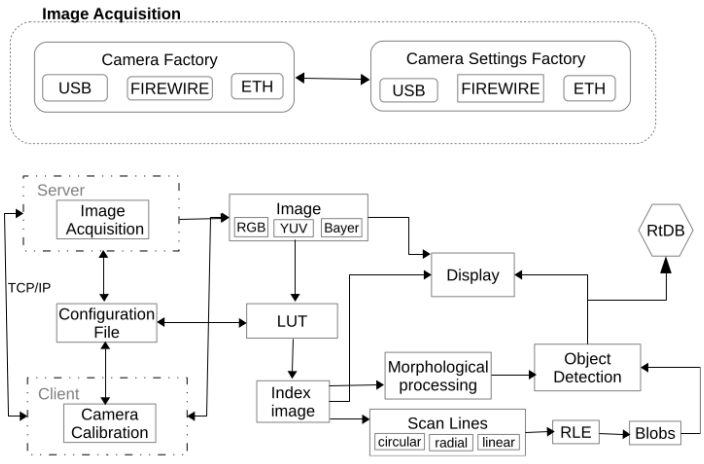

Figura 2.3: Biblioteca de visão (UAVision) para projetos de robótica UA a ser usada no projeto CAMBADA [7].

2.5 Base Station

A *Base Station* [9] é uma ferramenta, que durante o decorrer dos jogos, serve de intermediário entre os agentes da equipa e a *Referee Box*. Porém, o uso da *Base Station* não se limita apenas ao jogo. Esta possui um grande conjunto de ferramentas de teste, controlo e monitorização que são usadas pela equipa para desenvolvimento e testes. Utilizando a RTDB para difundir as ordens, é possível controlar o comportamento dos agentes através da *Base Station*, podendo dar sinais de jogo (ex.: Start e Stop), indicação da cor da equipa (Cyan ou Magenta), indicação da baliza (Azul ou Amarela) e indicação de *Roles* [9].

Na liga MSL as equipas não podem interferir no jogo. A *Base Station* permite monitorizar o jogo numa só janela, disponibilizando graficamente as informações que os agentes estão a disponibilizar na RTDB durante os jogos, sendo estas as seguintes: estado do jogo, posição da bola e dos robôs, *Roles* e comportamentos assumidos pelos agentes, visibilidade e posse de bola, cor da atual equipa, qual o meio campo da equipa e alguns pontos para *debug* [9]. A Figura 2.4 mostra a interface principal desta aplicação.

Figura 2.4: Interface disponibilizada pela *Base Station*.

2.6 Agente Robótico

O processo Agente é o responsável por interpretar os dados dos módulos sensoriais, tomar decisões comportamentais e trabalhar a coordenação com os outros robôs. Cada robô tem definido um identificador único que é lido pelo Agente e o que tiver o menor identificador assume a função de guarda-redes. Os restantes robôs alternam as suas funções dependendo do estado do jogo, posse de bola e posição da mesma. O processo do Agente é dividido em quatro fases: fusão sensorial, posicionamento estratégico, decisão e execução. Na fusão sensorial, a

informação recolhida pelos sensores do robô é fundida com a informação partilhada pelos colegas de equipa e também com o conhecimento anterior sobre os diversos objetos. Reunida toda a informação, é atualizado o *World State*, que será usado para decisão e coordenação de alto nível.

Na fase de posicionamento estratégico o robô tenta ocupar uma posição estratégica, que pode variar consoante a situação de jogo. Em lances de bola parada da própria equipa são usadas formações de *Setpieces* [10], que consistem em formações para definir as posições dos jogadores intervenientes nas jogadas deste tipo, criadas especificamente para cada *Setpiece*. Em situações de jogo aberto, com a bola a ser disputada pelas duas equipas, são usadas formações de *Free Play*. Cada formação define a posição no campo a ser tomada por cada posicionamento estratégico tendo em conta a posição da bola ao longo de todo o campo. Em lances de bola parada da equipa adversária, são usadas formações com posicionamento estratégico adaptativo [7] baseado em mapa de utilidades, que leva a um posicionamento mais dinâmico, tendo também em conta o posicionamento dos adversários e não apenas a bola.

Na fase de decisão o Agente assume um *Role*, conferindo ao robô a noção de atitude. Assim como o a função do guarda-redes referida acima é um *Role*, também o robô mais próximo da bola assume o *Role Striker*, perseguindo a bola tentando recuperá-la. Existem diversos *Roles* para diversas situações de jogo.

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

2.7 BDI

A arquitetura *Belief–Desire–Intention* (BDI) é uma arquitetura desenvolvida para a programação de agentes inteligentes. Podendo ser caracterizada pela implementação das crenças, desejos e intenções do agente, usa esses conceitos para resolver um problema particular em programação de agentes. Este fornece um mecanismo para separar a atividade de escolha de um plano da execução de planos ativos. Os agentes BDI são capazes de balancear o tempo despendido na deliberação sobre planos e a execução desses mesmos planos. A criação dos planos não faz parte do cariz do modelo, sendo da responsabilidade do programador.

A arquitetura de um sistema BDI é representada pela atitudes mentais já referidas: Crenças (*Beliefs*), Desejos (*Desires*) e Intenções (*Intentions*). As Crenças representam a informação do estado do agente e do mundo, ou seja, as suas crenças sobre o mundo, incluindo outros agentes. Os Desejos representam o estado motivacional do agente, objetivos ou situações que o agente quer alcançar. As Intenções representam o estado deliberado do agente, ou seja, aquilo que o agente escolheu fazer [11]. Na Figura 2.5 está representado um simples algoritmo de um idealizado intérprete de BDI.

O projeto CAMBADA usa uma arquitetura de agente do tipo BDI. Em cada *Role* existe uma lista de prioridades de opções (*Behaviours*) que podem ser executadas. O *Behaviour* a ser executado no momento é a Intenção do agente. Sempre que um *Role* é chamado, são verificadas as *Commitment Conditions* e as *Invocation Conditions* de todos os *Behaviours*. O *Role* comuta de *Behaviour* caso a *Commitment Condition* do *Behaviour* seja falsa ou a *Invocation Condition* de outro *Behaviour* de maior prioridade seja verdadeira.

1. initialize-state
2. repeat
    1. options: option-generator(event-queue)
    2. selected-options: deliberate(options)
    3. update-intentions(selected-options)
    4. execute()
    5. get-new-external-events()
    6. drop-unsuccessful-attitudes()
    7. drop-impossible-attitudes()
3. end repeat

Figura 2.5: Algoritmo para trabalho de equipa automatizado baseado em BDI [12].

Capítulo 3

Algoritmos para Controlo de Bola com o Corpo do Robô

Este capítulo vai abordar os comportamentos desenvolvidos para o controlo da bola, usando o corpo do robô para dar um ou vários embates na bola, de modo a que a bola se desloque para o ponto pretendido. Também será abordada a integração destes comportamentos no projeto CAMBADA.

Usando apenas o corpo do robô, pretende-se deslocar a bola para um ponto do campo (*BallTarget*). Para o embate do robô com a bola usa-se como superfície de embate as laterais planas do corpo do robô.

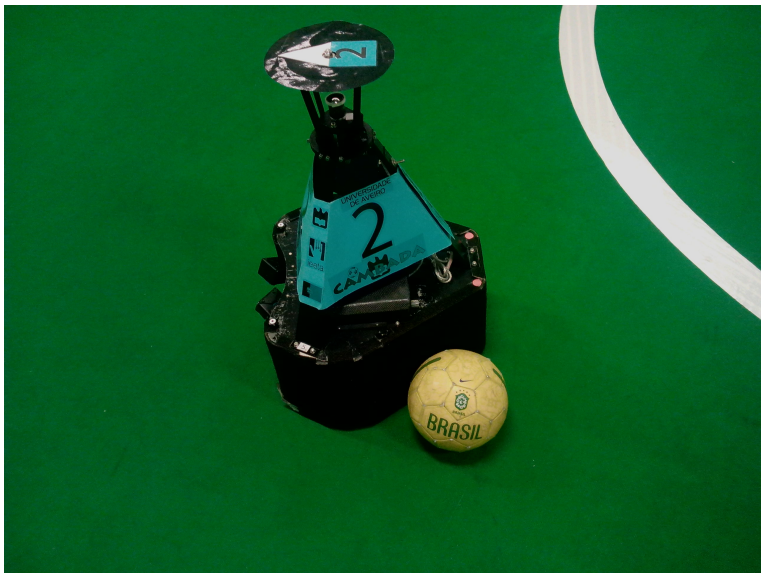

Figura 3.1: Bola na superfície de embate com o robô.

Para uma melhor abordagem do desafio, divide-se o trabalho por etapas, analisando primeiro casos com a bola parada, podendo ir à bola diretamente, percorrendo o caminho mais curto entre a posição inicial do robô e a bola ou ir à bola com correção da trajetória do robô. Em ambos os casos é necessário o cálculo da orientação do robô (*RobotOrientation*), ou seja, a orientação que o robô terá de ter para embater na bola com a lateral do seu corpo. São

também analisados casos com a bola em movimento, analisando as diferenças para os casos com a bola parada. É analisado um caso particular de bola em movimento, que é a tentativa de desviar a bola para a baliza na marcação de um pontapé de canto. Por fim é analisado a reposição de bola com o corpo do robô em jogo em lances de *Setpieces* quando o dispositivo ativo de *grabber* ou de *kicker* está avariado.

Todo o projeto CAMBADA está desenvolvido em Linux 32 bits, sendo este o ambiente de desenvolvimento deste projeto. A linguagem de programação usada para o desenvolvimento dos algoritmos é C++.

3.1 Situações de Bola Parada

Para o caso da bola parada foram desenvolvidos dois modelos, o Modelo Direto (3.1.1) que consiste em fazer deslocar o robô direto à bola e o Modelo com ajuste de trajetória (3.1.2). Para este último caso foram desenvolvidas três abordagens diferentes.

3.1.1 Modelo Direto

Este primeiro modelo consiste em fazer o robô ir direto à bola, sem fazer ajustes de posição, percorrendo o caminho mais curto entre a posição atual do robô e a posição da bola. Os robôs CAMBADA estão projetados para parar no ponto desejado (*RoboTarget*), então para que o robô chegue à bola com velocidade suficiente para a empurrar, o robô não pode ter o *RoboTarget* na bola mas sim após esta, na linha que atravessa o robô e a bola, forçando o robô a passar no ponto *RoboTarget* com velocidade. O modo como o robô se orienta para a bola é abordado em 3.1.3. O algoritmo para este modelo é apresentado em Algoritmo 1, que será adicionado ao algoritmo 5.

Algorithm 1 Modelo Direto

Input: DriveVector dv
Output: DriveVector dv

```
1: ballPosition = posição atual da bola
2: ownPosition = posição do robô
3: ballTarget = posição para onde se quer enviar a bola
4: ballToMe = ballPosition - ownPosition
5: increm = ballToMe / ballToMe.length()
6: if bola dentro de campo then
7:     RoboTarget = ballPosition + 2*increm
8: end if
9: ...
       [Algoritmo de orientação]
10: ...
11: dv = move(RoboTarget, RobotOrientation)
```

3.1.2 Modelo com Ajuste de Trajetória

Este modelo consiste em ajustar a trajetória do robô, provocando um melhor alinhamento desta com a bola e com o *BallTarget*. Para tal, foram desenvolvidas três abordagens diferentes, baseadas em pontos intermédios por onde o robô terá que passar, antes de se deslocar para a bola. Essas abordagens serão explicadas em 3.1.2.1, 3.1.2.2 e 3.1.2.3.

3.1.2.1 Abordagem 1

Este modelo consiste em obrigar o robô a passar na vizinhança de um ponto (*pti*) criado 2 metros antes da bola, na linha que atravessa a bola e o *BallTarget* desejado. Para tal considera-se que o *RoboTarget* é o ponto *pti*. Quando robô está a menos de 2 metros de *pti*, o *RoboTarget* passa a ser a bola. Novamente, para que o robô chegue à bola com velocidade suficiente para a empurrar, o robô não tem o *RoboTarget* na bola mas sim 2 metros após a bola, na linha que atravessa o robô e a bola. O modo como o robô se orienta para a bola é abordado em 3.1.3. Na Figura 3.2 pode-se ver representado o ponto *pti*, a zona de ataque direto à bola e a trajetória prevista. O algoritmo para este modelo é apresentado em Algoritmo 2, que será adicionado ao algoritmo 5.

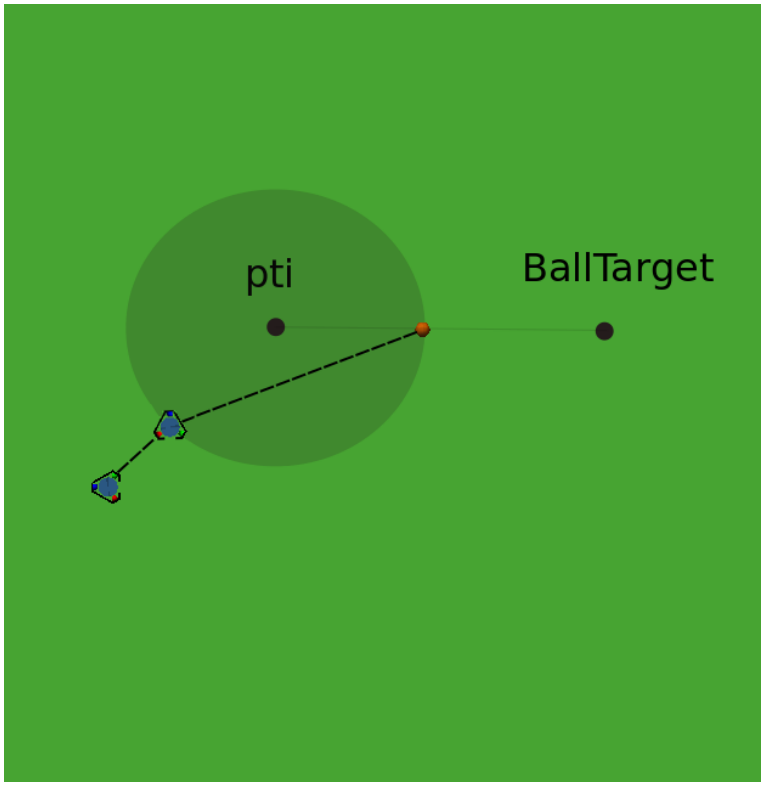

Figura 3.2: Representação do ponto *pti* e zona de ataque direto à bola.

3.1.2.2 Abordagem 2

À semelhança da Abordagem 1 (3.1.2.1), este modelo também obriga o robô a passar na vizinhança do ponto *pti* criado 2 metros antes da bola, na linha que atravessa a bola e o

Algorithm 2 Modelo com ajuste de trajetória - Abordagem 1

Input: DriveVector dv
Output: DriveVector dv

```
1: ballPosition = posição atual da bola
2: ownPosition = posição do robô
3: ballTarget = posição para onde se quer enviar a bola
4: ballToTarget = vetor entre o ballPosition e o ballTarget
5: ballToMe = ballPosition - ownPosition
6: increm = ballToMe / ballToMe.length()
7: vector = ballToTarget / ballToTarget.length();
8: pti = ballPosition + 2* vector.rotate_half();
9: ptiToMe = vetor entre o ownPosition e pti;
10: increm2 = ptiToMe / ptiToMe.length();
11: if bola dentro de campo then
12:     if ptiToMe.length() > 2.0 then
13:         RoboTarget = pti + 2*increm2
14:     else
15:         RoboTarget = ballPosition + 2*increm
16:     end if
17: end if
18: ...
        [Algoritmo de orientação]
19: ...
20: dv = move(RoboTarget, RobotOrientation)
```

BallTarget desejado. Quando robô está a menos de 2 metros de *pti* mas a mais de 0,5 metros da bola, são criados mais pontos intermédios sobre linha que atravessa a bola e o *BallTarget* desejado. Cada ponto é calculado usando a distância do robô à bola. O ponto é criado a uma distância da bola de 75% da distância do robô à bola. Quando o robô já está a menos de 0,5 metros da bola, o *RoboTarget* passa a ser 2 metros após a bola, na linha que atravessa o robô e a bola. O modo como o robô se orienta para a bola é abordado em 3.1.3. Na Figura 3.3 pode-se ver representado o ponto *pti*, os pontos intermédios, a zona de ataque direto à bola e a trajetória prevista. O algoritmo para este modelo é apresentado em Algoritmo 3, que será adicionado ao algoritmo 5.

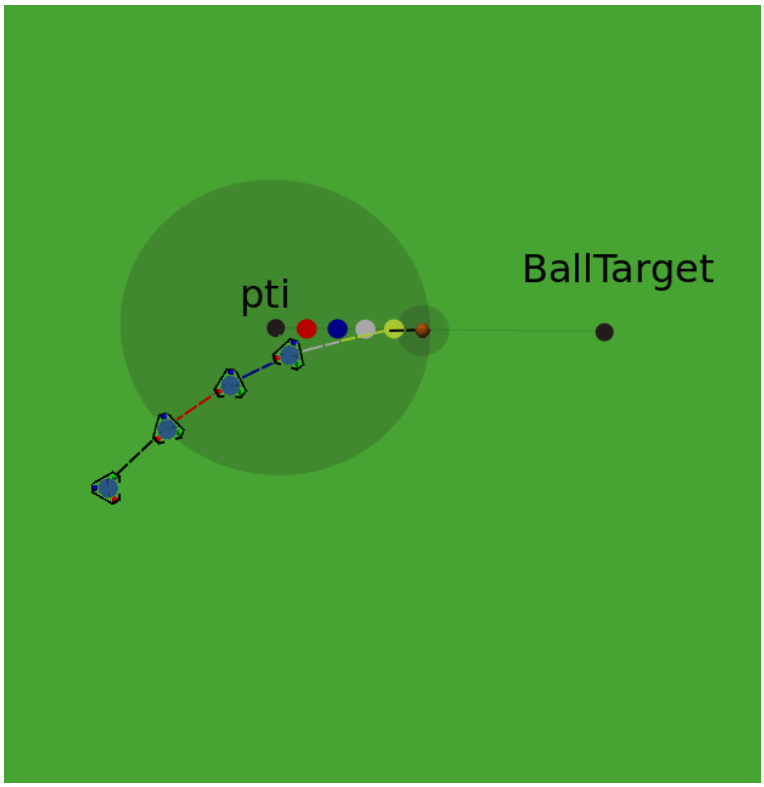

Figura 3.3: Representação do ponto *pti*, dos pontos intermédios e zona de ataque direto à bola.

3.1.2.3 Abordagem 3

Esta abordagem, tal como na Abordagem 1 (3.1.2.1) e na Abordagem 2 (3.1.2.2), também obriga o robô a passar na vizinhança do ponto *pti* criado 2 metros antes da bola, na linha que atravessa a bola e o *BallTarget* desejado. Foram criadas 3 zonas, que quando o robô se encontra dentro delas, passa a ter o *RoboTarget* 2 metros após a bola, na linha que atravessa o robô e a bola. O modo como o robô se orienta para a bola é abordado em 3.1.3. Na Figura 3.4 pode-se ver representado o ponto *pti*, as zonas de ataque direto à bola e a trajetória prevista. O algoritmo para este modelo é apresentado em Algoritmo 4, que será adicionado ao algoritmo 5.

Algorithm 3 Modelo com ajuste de trajetória - Abordagem 2

Input: DriveVector dv

Output: DriveVector dv

```
ballPosition = posição atual da bola
ownPosition = posição do robô
ballTarget = posição para onde se quer enviar a bola
ballToTarget = vetor entre o ballPosition e o ballTarget
ballToMe = ballPosition - ownPosition
increm = ballToMe / ballToMe.length()
vector = ballToTarget / ballToTarget.length()
pti = ballPosition + 2* vector.rotate_half()
ptiToMe = vetor entre o ownPosition e pti
increm2 = ptiToMe / ptiToMe.length()
if  bola dentro de campo  then
    if ptiToMe.length() > 2.0 then
        RoboTarget = pti + 2*increm2
    else if ptiToMe.length() > 2.0 e ballToMe.length() > 0.5 then
        vectorX = targetAbs - ballTarget
        vectorXnorm = vectorX / vectorX.length()
        lengthPti = (ballPosition - ownPosition).length()
        pti3 = (vectorXnorm * lengthPti)
        pti2 = ballPosition + pti3 * 0.75
        ptiToMe = pti2-ownPosition
        increm2 = ptiToMe / ptiToMe.length()
        RoboTarget = pti2 + 2*increm2
    else
        RoboTarget = ballPosition + 2*increm
    end if
end if
...
      [Algoritmo de orientação]
...
dv = move(RoboTarget, RobotOrientation)
```

Algorithm 4 Modelo com ajuste de trajetória - Abordagem 3

Input: DriveVector dv

Output: DriveVector dv

```
1: ballPosition = posição atual da bola
2: ownPosition = posição do robô
3: ballTarget = posição para onde se quer enviar a bola
4: ballToTarget = vetor entre o ballPosition e o ballTarget
5: ballToMe = ballPosition - ownPosition
6: increm = ballToMe / ballToMe.length()
7: point = ballToTarget / ballToTarget.length()
8: pti = ballPosition + 2* vector.rotate_half()
9: ptiToMe = vetor entre o ownPosition e pti
10: increm2 = ptiToMe / ptiToMe.length()
11: m = (ballTarget.y - ballPosition.y) / (ballTarget.x - ballPosition.x);
12: b = ballTarget.y - m * ballTarget.x;
13: expectedPoint.y = ownPosition.y;
14: expectedPoint.x = (expectedPoint.y - b) / m;
15: distancia = abs(ownPosition.x - expectedPoint.x);
16: if bola dentro de campo then
17:     if ( (ptiToMe.length() <= 2.2) e (((ballToMe.length() < 2.0) e (distancia < 1.0))
    ou ((ballToMe.length() >= 2.0) e (ballToMe.length() < 4.0) e (distancia < 2.0)) ou
    ((ballToMe.length() > 4.0) e (distancia < 3.0)))) then
18:         RoboTarget = BallPosition + 2*increm
19:     else
20:         RoboTarget = pti + 2*increm2
21:     end if
22: end if
23: ...
        [Algoritmo de orientação]
24: ...
25: dv = move(RoboTarget, RobotOrientation)
```

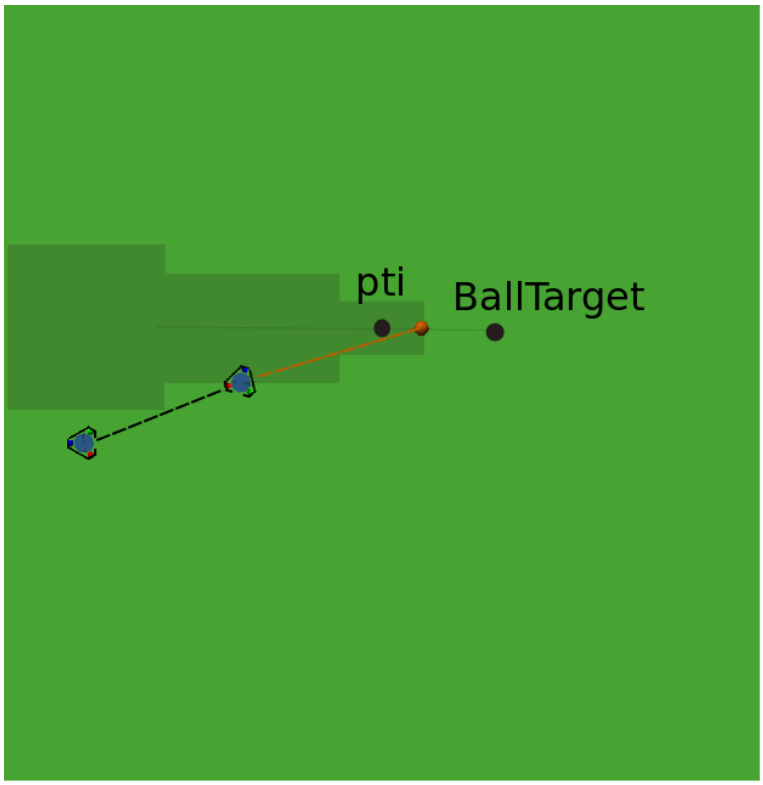

Figura 3.4: Representação do ponto *pti* e as zonas de ataque direto à bola.

3.1.3 Orientação Final do Robô

O que se pretende é usar as laterais planas robô, que tem têm orientações de 120^o para a frente do robô (Figura 3.5), para chocar com a bola, fazendo com que esta se desloque na trajetória desejada. Para tal é necessário que o robô tenha a orientação correta. Esta orientação é calculada com base na posição da bola na altura de embate e com o ponto de *BallTarget* desejado. Denomina-se por orientação do robô (*Orientation*) o vetor unitário criado com origem no centro do robô e extremidade na parte frontal do mesmo. Dado que os robôs têm um forma hexagonal assimétrica, os vetores perpendiculares aos planos das faces laterais do robô têm ângulos de 120° e $-120°$ com o vetor *Orientation* (Figura 3.5). Para representar esses vetores, roda-se o vetor *Orientation* em 120° e $-120°$, criando os vetores *OrientationMais120* e *OrientationMenos120*, respetivamente. Para descobrir a orientação correta, é criado um vetor com origem na posição da bola e com extremidade no *BallTarget* desejado, denominado *BallToTarget*. Calculando os ângulos entre o vetor *BallToTarget* e os vetores *OrientationMais120* e *OrientationMenos120*, obtém-se *AngleToLeft* e *AngleToRight*, respetivamente. Se o módulo do ângulo *AngleToLeft* for maior que o módulo do ângulo *AngleToRight*, obtém-se um vetor *TargetOrientation* que é o vetor *BallToTarget* rodado de 120°, caso contrário a rotação é de $-120°$. Para obter o ponto no campo para qual o robô deve estar orientado, o vetor *TargetOrientation* é somado à posição do robô, obtendo assim o vetor final de orientação *RobotOrientation*. O algoritmo para este modelo é apresentado em Algoritmo 5, que será usado juntamente com os algoritmos de cada modelo.

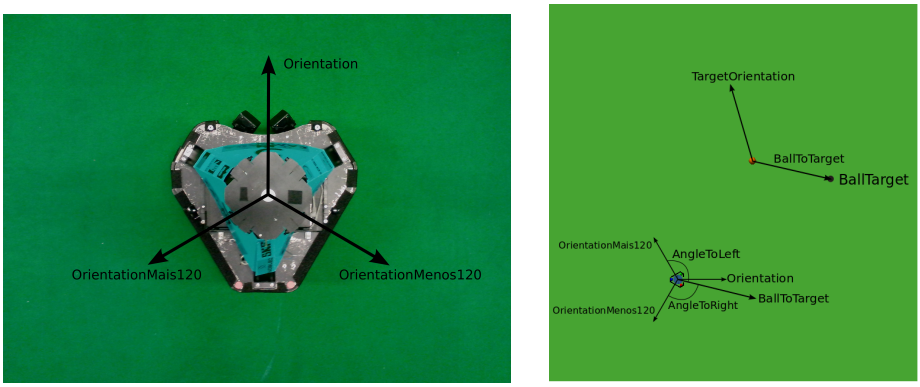

Figura 3.5: Representação dos vetores necessários ao cálculo da orientação.

Algorithm 5 Orientação final do robô

Input: DriveVector dv
Output: DriveVector dv

ballPosition = posição atual da bola
ownPosition = posição do robô
ballTarget = posição para onde se quer enviar a bola
ballToTarget = vetor entre o ballPosition e o ballTarget
Orientation = orientação da frente do robô
OrientationMenos120 = Orientation.rotate(-120^o)
OrientationMais120 = Orientation.rotate(120^o)
angleToLeft = ângulo entre os vetores OrientationMais120 e ballToTarget
angleToRight = ângulo entre os vetores OrientationMenos120 e ballToTarget
if |angleToLeft| > |angleToRight| **then**
    TargetOrientation = ballToTarget.rotate(120^o)
else
    TargetOrientation = ballToTarget.rotate(-120^o)
end if
RobotOrientation = ownPosition + TargetOrientation

3.2 Situações de Bola em Movimento

Para a bola em movimento, foram replicados os modelos de bola parada, com a diferença de não se usar a localização atual da bola, mas sim a posição de interseção do robô com a bola, fornecida através de uma função disponível no projeto CAMBADA. Esta função baseia-se nas velocidades e direções da bola e robô para fazer uma estimativa para o ponto de interseção do robô e com a bola. Para a orientação do robô é usado o modelo Orientação final do robô (3.1.3). Estes modelos de bola em movimento ainda se encontram em fase de afinações, não produzindo ainda resultados aceitáveis.

Um caso de bola em movimento mas com condições específicas é o caso da marcação de cantos, em que se sabe à partida de onde a bola virá e para onde será passada, obtendo a linha de passe, que consiste na trajetória que a bola irá ter.

3.2.1 Cantos

Para os cantos, foi necessário rever a orientação do robô, pois a bola está em movimento. O objetivo é parar o robô na trajetória da bola, para que esta choque contra ele e siga para a baliza. Tal efeito é considerado um choque elástico [13].

Sendo este um caso de um choque elástico, para que a bola saia do embate na orientação desejada, a superfície de embate tem que estar orientada segundo a bissetriz do ângulo obtido entre o vetor com origem na posição do robô e extremidade na posição inicial da bola e o vetor com origem na posição do robô e extremidade no *BallTarget* [13].

Para o robô poder parar na trajetória da bola, o robô que marca o canto fornece a trajetória da bola, permitindo ao primeiro robô que se desloca para esta. A Figura 3.6 representa os vetores *RoboToBall*, *RoboToBallTarget* e a bissetriz do ângulo entre esses vetores. O algoritmo para este modelo é apresentado em Algoritmo 6.

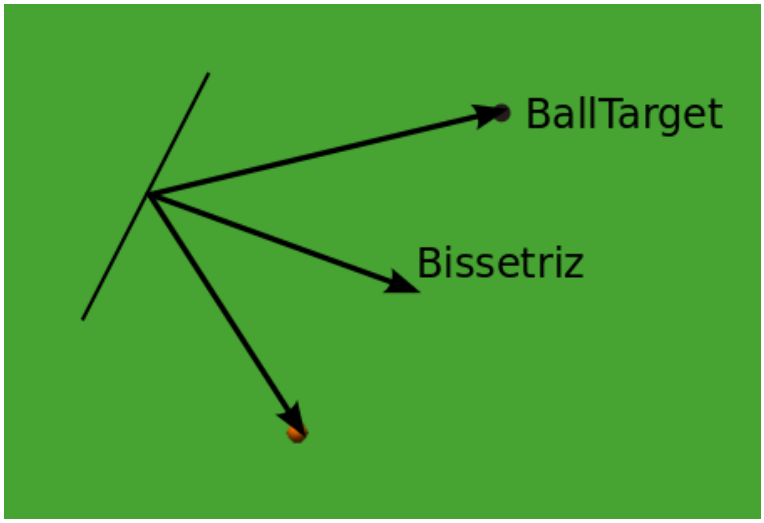

Figura 3.6: Representação dos vetores *RoboToBall*, *RoboToBallTarget* e a bissetriz do ângulo entre esses vetores.

3.3 Setpieces

Para contornar uma avaria mecânica no dispositivo de *grabber* ou de *kicker* na marcação de uma bola parada, foi desenvolvido um comportamento que consiste em passar a bola com o corpo do robô a um colega de equipa. Este comportamento faz com que o robô comece por por se afastar ligeiramente da bola. Este afastamento serve para ter algum espaço que lhe permita posteriormente avançar para a bola atingindo velocidade suficiente para fazer deslocar a bola. Durante o afastamento, o robô alinha uma das laterais do robô com a bola e o colega de equipa a quem vai passar a bola, segundo o modelo de Orientação final do robô (3.1.3), sendo que o ponto *BallTarget* é a localização do colega de equipa. Quando a lateral do robô está alinhada com a bola e o colega de equipa, o robô avança para a bola, parando quando toca na bola. O algoritmo para este modelo é apresentado em Algoritmo 7, que será usado juntamente com os algoritmo 5.

Algorithm 6 Behaviour - Cantos com o corpo do robô

Input: RoleReceiver parent, DriveVector dv
Output: DriveVector dv

```
1: ballPosition = posição atual da bola
2: ballTarget = posição para onde se quer enviar a bola
3: passLine = trajetória da bola
4: ownPosition = posição do robô
5: nearPoint = passLine.perpendicular_point(ownPosition)
6: ballInLine = passLine.perpendicular_point(ballPosition)
7: line2ball = ballPos - ballInLine
8: RoboTarget = nearPoint + line2ball
9: ballToTargetAbs = passLine.p1 - ownPosition [ p1 é 1º ponto da passLine ]
10: ballTargetToBall = ballTarget - ownPosition;
11: bissectorVec = vetor segundo a bissetriz do ângulo obtido entre os vetores ballToTarge-
    tAbs e ballTargetToBall
12: increm = bissectorVec / bissectorVec.length()
13: bissectorVec += increm
14: Orientation = orientação da frente do robô
15: OrientationMenos120 = robotOrientation.rotate( −120º )
16: OrientationMais120 = robotOrientation.rotate( 120º )
17: angleToLeft = ângulo entre o vetor bissectorVec e o vetor OrientationMais120
18: angleToRight = ângulo entre o vetor bissectorVec e o vetor OrientationMenos120
19: if |angleToLeft| > |angleToRight| then
20:     targetOrientation = bissectorVec.rotate( 120º )
21: else
22:     targetOrientation = bissectorVec.rotate( −120º )
23: end if
24: RobotOrientation = ownPosition + targetOrientation;
25: if bola passada para mim then
26:     dv = move(RoboTarget,RobotOrientation)
27: else
28:     dv = move(RoboTarget,RobotOrientation)
29:     dv->velX = 0.0
30:     dv->velY = 0.0
31: end if
```

Algorithm 7 Behaviour - Reposição da bola em jogo usando o corpo do robô

Input: RoleReplacer parent, DriveVector dv
Output: DriveVector dv

```
1: receiverPos = posição para onde irá passar bola
2: ballPosition = posição atual da bola
3: ballTargetToBall = receiverPos - ballPosition
4: ownPosition = posição do robô
5: RoboTarget = ballPosition + (ballPosition - ownPosition).setLength(1.0)
6: position = ponto 0.7 metros atrás do robô
7: ...
       [Algoritmo de orientação]
8: ...
9: if | targetOriVec.angle() - robotOrientation.angle() | > 5 then
10:    dv=move(position,RobotOrientation)
11: else
12:    dv=move(RoboTarget,RobotOrientation)
13: end if
14: if distancia do robô à bola < 1.0 then [parar quando o robô chega à posição da bola]
15:    parentRole->ballPassed = true
16: end if
```

3.4 Integração no CAMBADA

Nesta fase, apenas os modelos de Cantos (3.2.1) e de Setpieces (3.3) se encontram integrados no projeto CAMBADA, tendo sido efetuadas algumas alterações no Agente para permitir essas integrações.

No caso dos Cantos, foram criados na ferramenta de configuração alguns campos que permitem ativar/desativar este tipo de jogada, assim como configurar o ponto para onde a bola é passada, configurar a distância inicial do robô à linha de passe e configurar o *offset* do *BallTarget* em relação ao centro da baliza. Esta ferramenta usada é uma ferramenta disponível no projeto CAMBADA e que serve para configurar vários aspetos relacionados com o projeto. É nesta ferramenta que se pode configurar as posições estratégicas usadas em lances de *Setpieces*, as características do campo a ser usado, alguns parâmetros usados nos *Behaviours* e os valores dos parâmetros dos controladores dos motores.

O comportamento dos Cantos foi adicionado à lista de *Behaviours* executáveis do *RoleReceiver*, referida em 2.7, sendo acionado apenas pelo robô que se encontra mais perto da baliza contrária.

No caso das *Setpieces*, o comportamento foi adicionado à lista de *Behaviours* executáveis do *RoleReplacer*, entrando em funcionamento apenas quando o robô está em modo *Handicap*. O modo *Handicap* é um modo acionado no PC do próprio robô, alterando o ficheiro de configuração chamado *HandicapGrabber*, indicando que o robô está com problemas no dispositivo de *grabber* ou de *kicker*.

Capítulo 4

Resultados Experimentais

Na sequência do desenvolvimento dos modelos foram efetuados testes para comprovar o seu funcionamento. A análise destes testes possibilitam a deteção de erros e a realização de melhorias aos respetivos algoritmos. Neste Capítulo abordam-se estes testes, realizados com recurso a vários pontos no campo, pontos esses representados na Figura 4.1. Os pontos a azul representam várias posições iniciais do robô, os pontos a vermelho representam as várias posições iniciais da bola e os dois pontos a preto representam os pontos utilizados como *BallTarget*.

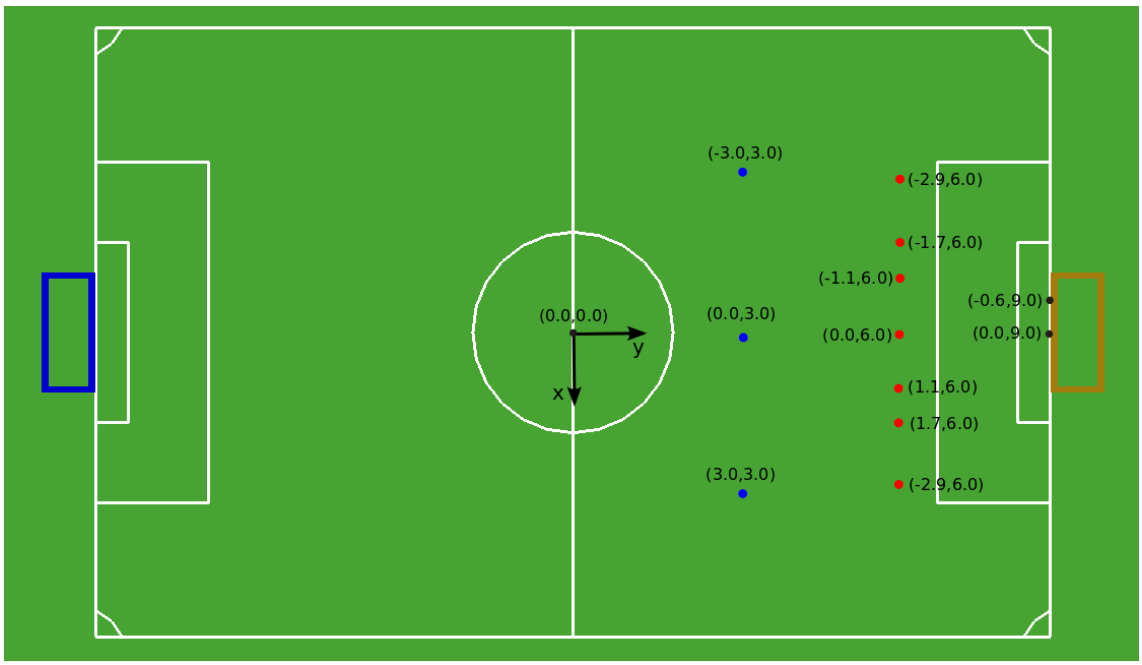

Figura 4.1: Pontos usados nos testes efetuados.

4.1 Bola Parada

Para os vários modelos de bola parada desenvolvidos, os testes consistem em colocar o robô e a bola em posições fixas, variando depois a posição da bola e do robô. Para cada caso

foram realizadas 10 repetições, medindo o desvio sobre a linha final (y=9.0) em relação ao *BallTarget*(0.0,9.0), calculando posteriormente a média desse desvio. Para todos os casos o *BallTarget* será o centro da baliza contrária, tendo as seguintes coordenadas (0.0,9.0).

4.1.1 Modelo Direto

Neste modelo foram realizados testes com o robô em 3 posições iniciais diferentes, sendo essas (-3.0,4.0), (0.0,4.0) e (3.0,4.0), obtendo para várias posições da bola os resultados apresentados nas tabelas 4.1, 4.2 e 4.3, respetivamente. Analisando as tabelas referidas anteriormente, verifica-se que este modelo produz melhores resultados quando o robô está alinhado com a bola e o *BallTarget*. Para casos de desalinhamento extremo, este modelo revela-se ineficaz, produzindo desvios por vezes superiores a 2 metros. Na Tabela 4.2 pode-se verificar uma anomalia, pois para a bola colocada em (0.0,6.0), estando o robô em (0.0,4.0) e perfeitamente alinhado com a bola e o *BallTarget*, o desvio médio é 0.93m, quando o esperado seria um desvio muito inferior a este. Tal fenómeno pode ser explicado pela dificuldade conhecida de o robô CAMBADA fazer rotação de orientação enquanto se desloca em linha reta, provocando um desvio da trajetória desejada. A representação desta anomalia pode ser observada na Figura 4.2, em que a linha a tracejado representa a trajetória que seria esperada e a linha a cheio representa a trajetória real. Um vídeo demonstrativo pode ser observado em http://youtu.be/86UDkJp4NJo.

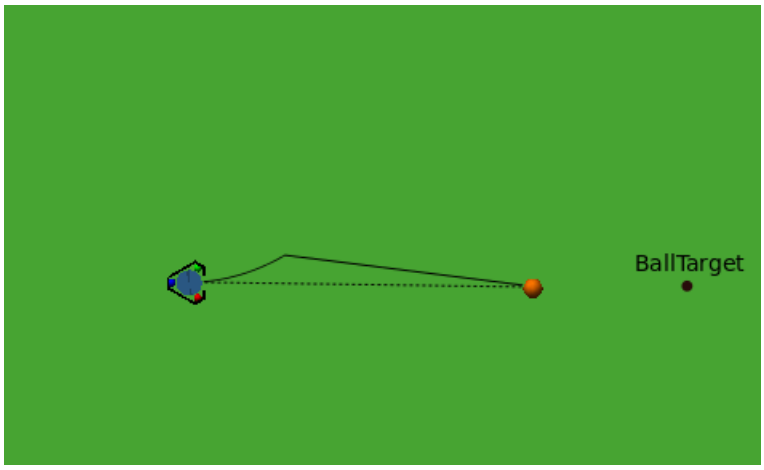

Figura 4.2: Representação da anomalia observada em alinhamento perfeito.

	Posição Inicial da Bola												
	(-2.9,6.0)			(-2.3,6.0)			(-1.7,6.0)			(-1.1,6.0)		(-0.6,6.0)	
	x	$\|x\|$	$\overline{\|x\|}$	x	$\|x\|$	$\overline{\|x\|}$	x	$\|x\|$	$\overline{\|x\|}$	x	$\overline{x}$	x	$\overline{x}$
1	-2.12	2.12		-0.67	0.67		0.11	0.11		0.89		1.44	
2	-2.21	2.21		0.72	0.72		0.13	0.13		0.82		0.93	
3	-2.44	2.44		-0.43	0.43		-0.10	0.10		0.41		1.21	
4	-1.56	1.56		-0.67	0.67		-0.17	0.17		0.44		0.78	
5	-2.11	2.11	2.02	-0.78	0.78	0.54	1.72	1.72	0.38	0.56	0.48	1.84	1.76
6	-1.64	1.64		-0.30	0.30		0.14	0.14		0.32		2.12	
7	-2.63	2.63		-0.52	0.52		-0.89	0.89		0.14		0.84	
8	-1.52	1.52		-0.34	0.34		0.23	0.23		0.23		3.55	
9	-2.28	2.28		-0.41	0.41		0.03	0.03		0.67		2.20	
10	-1.71	1.71		-0.56	0.56		0.23	0.23		0.33		2.75	

Tabela 4.1: Modelo Direto com robô em (-3.0,4.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola											
	(-1.1,6.0)			(-0.6,6.0)			(0.0,6.0)		(0.6,6.0)		(1.1,6.0)	
	x	$\|x\|$	$\overline{\|x\|}$	x	$\|x\|$	$\overline{\|x\|}$	x	$\overline{x}$	x	$\overline{x}$	x	$\overline{x}$
1	-2.56	2.56		-0.42	0.42		0.91		0.1		2.14	
2	-2.44	2.44		-0.39	0.39		0.78		1.34		1.78	
3	-2.78	2.78		-1.13	1.13		0.74		0.22		2.03	
4	-1.39	1.39		-0.56	0.56		0.91		0.08		3.73	
5	-2.78	2.78	2.38	-0.64	0.64	0.79	0.78	0.93	0.32	0.42	1.55	2.02
6	-2.21	2.21		-1.21	1.21		1.24		0.57		1.89	
7	-2.63	2.63		-0.67	0.67		1.02		0.53		1.71	
8	-2.32	2.32		-0.73	0.73		0.86		0.43		1.88	
9	-2.37	2.37		-1.08	1.08		1.17		0.28		1.78	
10	-2.28	2.28		-1.03	1.03		0.84		0.36		1.67	

Tabela 4.2: Modelo Direto com robô em (0.0,4.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola												
	(0.6,6.0)			(1.1,6.0)			(1.7,6.0)			(2.3,6.0)		(2.9,6.0)	
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$	x	$\overline{x}$
1	-0.56	0.56		-0.07	0.07		0.24	0.24		0.29		2.29	
2	-0.63	0.63		-0.08	0.08		0.11	0.11		0.31		2.26	
3	-0.79	0.79		-0.21	0.21		-0.69	0.69		0.56		2.41	
4	-0.61	0.61		-0.24	0.24		0.45	0.45		0.24		2.34	
5	-0.98	0.98	0.69	-0.18	0.18	0.17	-0.68	0.68	0.48	0.45	0.48	2.13	2.15
6	-0.73	0.73		-0.07	0.07		-0.71	0.71		0.09		2.08	
7	-0.67	0.67		-0.13	0.13		-0.70	0.70		0.83		1.74	
8	-0.59	0.59		-0.27	0.27		-0.43	0.43		0.87		2.52	
9	-0.57	0.57		-0.28	0.28		0.31	0.31		0.51		1.28	
10	-0.72	0.72		-0.21	0.21		0.46	0.46		0.62		2.41	

Tabela 4.3: Modelo Direto com robô em (3.0,4.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

4.1.2 Modelo com ajuste de trajetória

Tal como no Modelo Direto 4.1.1, para cada abordagem foram realizados testes para várias posições da bola, com o robô em 3 posições iniciais diferentes, sendo essas (-3.0,3.0), (0.0,3.0) e (3.0,3.0).

4.1.2.1 Abordagem 1

Nesta abordagem, os resultados obtidos para várias posições da bola são apresentados nas tabelas 4.4, 4.5 e 4.6, respetivamente. Analisando as tabelas referidas anteriormente, verifica-se uma redução do desvio médio, quando comparado com o Modelo Direto 4.1.1. Quando a bola se encontra em (-2.9,6.0) e (-1.1,6.0) da Tabela 4.4, em (-1.1,6.0) e (1.1,6.0) da Tabela 4.5 e em (1.1,6.0) e (2.9,6.0) da Tabela 4.6, o desvio médio não ultrapassa 0.25m e o maior desvio registado foi 0.79m. Quando a bola se encontra em (1.1,6.0) da Tabela 4.4 e em (-1.1,6.0) da Tabela 4.6, o desvio médio registado é superior ao dos casos abordados anteriormente, sendo de 0.84m e de 0.82m, respetivamente. Tal facto pode ser explicado pelo não tão correto alinhamento do robô com a bola e o *BallTarget*, característica observada nesta Abordagem para pontos mais distantes da posição inicial do robô. Na Tabela 4.5 existe um caso semelhante ao observado no Modelo Direto 4.1.1, que é para a bola em (0.0,6.0). A explicação para esta anomalia é referida no caso do Modelo Direto 4.1.1. Um vídeo demonstrativo pode ser observado em http://youtu.be/qHVciyc1d0I.

	Posição Inicial da Bola							
	(-2.9,6.0)			(-1.1,6.0)			(1.1,6.0)	
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$
1	-0.22	0.22		0.11	0.11		0.62	
2	0.10	0.10		-0.07	0.07		1.01	
3	-0.33	0.33		-0.01	0.01		0.67	
4	-0.41	0.41		-0.05	0.05		0.76	
5	-0.29	0.29	0.25	0.13	0.13	0.06	0.80	0.84
6	-0.13	0.13		0.08	0.08		0.71	
7	-0.27	0.27		0.01	0.01		0.91	
8	-0.25	0.25		0.12	0.12		1.03	
9	-0.26	0.26		-0.01	0.01		0.89	
10	-0.26	0.26		0.04	0.04		0.96	

Tabela 4.4: Modelo com ajuste de trajetória (Abordagem 1) com robô em (-3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola					
	(-1.1,6.0)		(0.0,6.0)		(1.1,6.0)	
	x	$\overline{x}$	x	$\overline{x}$	x	$\overline{x}$
1	0.01		0.62		0.02	
2	0.07		0.82		0.12	
3	0.50		0.57		0.17	
4	0.79		0.49		0.15	
5	0.21	0.24	0.64	0.74	0.30	0.14
6	0.12		0.82		0.15	
7	0.16		0.79		0.13	
8	0.40		0.79		0.13	
9	0.06		0.61		0.18	
10	0.09		1.28		0.09	

Tabela 4.5: Modelo com ajuste de trajetória (Abordagem 1) com robô em (0.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola							
	(-1.1,6.0)			(1.1,6.0)		(2.9,6.0)		
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$
1	-0.44	0.44		0.24		-0.07	0.07	
2	-1.23	1.23		0.26		0.18	0.18	
3	-0.60	0.60		0.32		0.18	0.18	
4	-0.87	0.87		0.16		0.19	0.19	
5	-1.05	1.05	0.82	0.19	0.23	0.18	0.18	0.21
6	-0.69	0.69		0.30		0.10	0.10	
7	-1.07	1.07		0.12		0.44	0.44	
8	-1.05	1.05		0.25		0.16	0.16	
9	-0.63	0.63		0.20		0.14	0.14	
10	-0.53	0.53		0.23		0.41	0.41	

Tabela 4.6: Modelo com ajuste de trajetória (Abordagem 1) com robô em (3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

4.1.2.2 Abordagem 2

Nesta abordagem, os resultados obtidos para várias posições da bola são apresentados nas tabelas 4.7, 4.8 e 4.9, respetivamente. Analisando as tabelas referidas anteriormente, verifica-se um melhoramento do desvio médio para casos em que a bola se encontra mais distante da posição inicial do robô, quando comparado com a Abordagem 1 4.1.2.1. Estes casos podem ser observados quando a bola se encontra na posição (1.1,6.0) da Tabela 4.7 e na posição (-1.1,6.0) da Tabela 4.9. No entanto, para posições onde a Abordagem 1 obtinha melhores resultados, esta Abordagem produz resultados piores, como observado nos casos da bola na posição (-1.1,6.0) da Tabela 4.7, (-1.1,6.0) da Tabela 4.8 e (1.1,6.0) e (2.9,6.0) da Tabela 4.9. Tal como observado no Modelo Direto 4.1.1 e na Abordagem 1 4.1.2.1, também nesta abordagem existe uma anomalia, para o caso da bola na posição (0.0,6.0) da Tabela 4.8. A explicação para esta anomalia é referida no caso do Modelo Direto 4.1.1. Um vídeo demonstrativo pode ser observado em http://youtu.be/6Xi9yqMdOA8.

	Posição Inicial da Bola											
	(-2.9,6.0)			(-1.1,6.0)			(0.0,6.0)			(1.1,6.0)		
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$
1	0.26	0.26		1.19	1.19		-0.06	0.06		-0.28	0.28	
2	-0.37	0.37		1.25	1.25		0.08	0.08		-0.10	0.10	
3	0.02	0.02		-0.19	0.19		-0.05	0.05		-0.30	0.30	
4	0.55	0.55		-0.14	0.14		0.00	0.00		-0.50	0.50	
5	-0.20	0.20	0.25	0.99	0.99	0.92	0.03	0.03	0.05	-0.31	0.31	0.22
6	0.34	0.34		-0.21	0.21		0.02	0.02		-0.22	0.22	
7	0.03	0.03		1.57	1.57		0.03	0.03		-0.06	0.06	
8	0.46	0.46		1.29	1.29		-0.10	0.10		-0.12	0.12	
9	-0.10	0.10		0.79	0.79		-0.03	0.03		-0.19	0.19	
10	0.20	0.20		1.62	1.62		-0.09	0.09		-0.14	0.14	

Tabela 4.7: Modelo com ajuste de trajetória (Abordagem 2) com robô em (-3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola						
	(-1.1,6.0)			(0.0,6.0)		(1.1,6.0)	
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$	x	$\overline{x}$
1	0.01	0.01		0.83		0.02	
2	-0.19	0.19		1.02		0.12	
3	0.68	0.68		0.62		0.17	
4	0.67	0.67		0.82		0.15	
5	0.51	0.51	0.55	0.57	0.72	0.30	0.14
6	0.06	0.06		0.45		0.15	
7	1.12	1.12		0.64		0.13	
8	1.04	1.04		0.82		0.08	
9	0.63	0.63		0.79		0.18	
10	0.58	0.58		0.61		0.09	

Tabela 4.8: Modelo com ajuste de trajetória (Abordagem 2) com robô em (0.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola										
	(-1.1,6.0)			(0.0,6.0)		(1.1,6.0)			(2.9,6.0)		
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$
1	-0.03	0.03		0.02		-0.43	0.43		-0.34	0.34	
2	0.12	0.12		0.09		-0.75	0.75		-0.77	0.77	
3	0.14	0.14		0.24		-0.31	0.31		-0.36	0.36	
4	-0.06	0.06		0.13		-0.57	0.57		-0.34	0.34	
5	-0.03	0.03	0.14	0.05	0.12	-0.36	0.36	0.51	0.99	0.99	0.50
6	0.35	0.35		0.21		-0.57	0.57		-0.75	0.75	
7	0.01	0.01		0.12		-0.48	0.48		-0.08	0.08	
8	-0.30	0.30		0.11		-0.50	0.50		0.23	0.23	
9	0.26	0.26		0.18		-0.39	0.39		-0.76	0.76	
10	0.05	0.05		0.07		-0.70	0.70		-0.42	0.42	

Tabela 4.9: Modelo com ajuste de trajetória (Abordagem 2) com robô em (3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

4.1.2.3 Abordagem 3

Nesta abordagem, os resultados obtidos para várias posições da bola são apresentados nas tabelas 4.10, 4.11 e 4.12, respetivamente. Tal como observado no Modelo Direto 4.1.1, na Abordagem 1 4.1.2.1 e na Abordagem 2 4.1.2.2, também nesta abordagem existe uma anomalia, para o caso da bola na posição (0.0,6.0) da Tabela 4.8. A explicação para esta anomalia é referida no caso do Modelo Direto 4.1.1. Tirando o caso da anomalia, esta Abordagem apresenta resultados mais coerentes entre as diversas posições da bola, tendo como máximo de desvio médio de 0.36m para o caso da bola na posição (-1.1,6.0) na Tabela 4.12, obtendo o desvio máximo de 0.55m para a bola na posição (-2.9,6.0) na Tabela 4.10. Um vídeo demonstrativo pode ser observado em http://youtu.be/EkPxhaRjQ28.

	Posição Inicial da Bola								
	(-2.9,6.0)		(-1.1,6.0)			(0.0,6.0)		(1.1,6.0)	
	x	$\overline{x}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$	x	$\overline{x}$
1	0.29		-0.05	0.05		0.21		0.33	
2	0.42		-0.04	0.04		0.25		0.19	
3	0.29		-0.03	0.03		0.29		0.27	
4	0.22		-0.06	0.06		0.13		0.25	
5	0.16	0.26	0.07	0.07	0.05	0.11	0.23	0.26	0.31
6	0.39		-0.07	0.07		0.23		0.35	
7	0.19		0.06	0.06		0.24		0.36	
8	0.55		0.00	0.00		0.20		0.42	
9	0.09		-0.03	0.03		0.32		0.42	
10	0.01		-0.04	0.04		0.31		0.28	

Tabela 4.10: Modelo com ajuste de trajetória (Abordagem 3) com robô em (-3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola							
	(-1.6,6.0)			(0.0,6.0)			(1.6,6.0)	
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\overline{x}$
1	-0.38	0.38		0.68	0.68		0.18	
2	-0.23	0.23		0.92	0.92		0.28	
3	-0.21	0.21		0.48	0.48		0.16	
4	-0.23	0.23		1.62	1.62		0.22	
5	-0.26	0.26	0.28	0.55	0.55	0.72	0.24	0.18
6	-0.22	0.22		0.58	0.58		0.07	
7	-0.35	0.35		0.66	0.66		0.22	
8	-0.33	0.33		-0.39	0.39		0.12	
9	-0.37	0.37		0.69	0.69		0.08	
10	-0.19	0.19		0.62	0.62		0.18	

Tabela 4.11: Modelo com ajuste de trajetória (Abordagem 3) com robô em (0.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

	Posição Inicial da Bola											
	(-1.1,6.0)			(0.0,6.0)			(1.1,6.0)			(2.9,6.0)		
	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$	x	$\lvert x\rvert$	$\overline{\lvert x\rvert}$
1	-0.29	0.29		-0.32	0.32		-0.03	0.03		-0.37	0.37	
2	-0.41	0.41		-0.31	0.31		0.07	0.07		-0.39	0.39	
3	-0.30	0.30		-0.36	0.36		0.06	0.06		-0.05	0.05	
4	-0.45	0.45		-0.25	0.25		0.10	0.10		-0.31	0.31	
5	-0.38	0.38	0.36	-0.29	0.29	0.29	0.09	0.09	0.07	-0.11	0.11	0.33
6	-0.29	0.29		-0.28	0.28		0.05	0.05		-0.47	0.47	
7	-0.28	0.28		-0.28	0.28		-0.03	0.03		-0.51	0.51	
8	-0.38	0.38		-0.38	0.38		0.11	0.11		-0.09	0.09	
9	-0.36	0.36		-0.22	0.22		0.04	0.04		-0.48	0.48	
10	-0.44	0.44		-0.24	0.24		0.12	0.12		-0.49	0.49	

Tabela 4.12: Modelo com ajuste de trajetória (Abordagem 3) com robô em (3.0,3.0) - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(0.0, 9.0)$.

4.2 Bola em movimento

Como descrito em 3.2, apenas o modelo para os cantos se encontra funcional, sendo que apenas para esse foram realizados testes. Estes consistem na marcação de 10 cantos nas mesmas condições.

4.2.1 Cantos

Para os testes realizados o robô encarregue de desviar a bola para a baliza encontra-se na posição (-0.3,6.4) e o robô encarregue de passar a bola, faz o passe para a posição (-5.5,6.5). Neste caso, o *BallTarget* é a posição (-0.6,6.0), dado que o canto foi marcado próximo da posição (6.0,9.0).

Analisando os resultados obtidos, apresentados na Tabela 4.13, pode se verificar que desvio médio da bola em relação ao *BallTarget* é de 0.44m, tendo um desvio máximo de 1.53m. Para este desvio de 1.53m, em que a bola passou a linha final na coordenada $x = -2.13m$, a bola falha claramente a baliza (entre x=-1.0 e x=1.0). Este desvio deve-se ao facto de o robô não se ter conseguido posicionar sobre a linha de passe em tempo útil, o que levou a bola a não embater na lateral do corpo do robô. Ignorando este valor, pode-se obter um novo desvio médio de 0.32m, sendo que o desvio máximo é de 0.64m. Um vídeo demonstrativo pode ser observado em http://youtu.be/MziduejOtw8.

x	$\lvert x-(-0.6)\rvert$	$\overline{\lvert x-(-0.6)\rvert}$
-0.54	0.06	
-0.77	0.17	
-0.25	0.35	
-0.73	0.13	
0.04	0.64	0.44
-0.17	0.43	
-0.71	0.11	
-2.13	1.53	
0.07	0.67	
-0.89	0.29	

Tabela 4.13: Cantos - Desvio da bola (m) em x (para $y = 9.0$) em relação ao *BallTarget*$(-0.6, 9.0)$.

4.3 Setpieces

Para o caso da reposição da bola em jogo em situações de *Setpieces* usando o corpo do robô, foram comparadas 10 reposições de bola em jogo, inicialmente com o robô no seu estado normal, utilizando os dispositivos de *grabber* e de *kicker* e posteriormente com o robô em modo *Handicap*. Nesta comparação foi registado o número de vezes que um segundo robô em *RoleReceiver* recebia a bola "à primeira", ou seja, conseguia "encaixar"a bola no dispositivo de *grabber*. A situação da bola estar "encaixada"no dispositivo de *grabber* é denominada por bola *engaged*.

Analisando a Tabela 4.14 é possível verificar que em ambos os casos o robô em *RoleReceiver* conseguiu ter a bola *engaged* "à primeira"em 7 ocasiões. Este resultado permite concluir que este modelo funciona bem, conseguindo fazer a reposição com a mesma percentagem de sucesso que um robô em estado normal. Um vídeo demonstrativo pode ser observado em http://youtu.be/ZFGPDHOL2yI.

	Normal	*Handicap*
Tentativas	10	10
Bola *Engaged*	7	7

Tabela 4.14: Marcação de *Setpieces* com o robô no estado normal e em modo *Handicap*.

Capítulo 5

Conclusão e trabalho futuro

Esta dissertação tinha por objetivo o desenvolvimento de comportamentos de controlo de bola usando o corpo do robô CAMBADA. Para atingir este objetivo, foram idealizados alguns comportamentos, quer para casos de bola parada quer para casos de bola em movimento. Para a bola parada, foram desenvolvidos modelos de trajetória, quer seja direto à bola ou ajustando a posição do robô para um melhor alinhamento com a bola e o ponto para onde se quer enviar a bola, forçando o robô a passar na vizinhança de pontos auxiliares. Para o modelo de ajuste de posição foram criadas 3 abordagens distintas.

Para a bola em movimento, foram replicados os modelos de bola parada, com a diferença da forma como é calculada a posição da bola, sendo neste caso usada a estimativa da interseção do robô com a bola.

Os testes efetuados a estes modelos permitem concluir que embora alguns produzam resultados aceitáveis, como o caso do modelo de Cantos e o caso do modelo de *Setpieces*, os modelos de Bola Parada podem ser melhorados. Para os modelos de bola em movimento, não há resultados a apresentar, pois os modelos precisam de ser afinados.

O resultado final é a criação de vários modelos, dois deles (cantos e *Setpieces*) funcionais e integrados no projeto CAMBADA. O modelo de cantos é configurável via ferramenta de configuração e pode ser desativado. O controlo da bola usando o corpo do robô CAMBADA não pode ser considerado um projeto acabado, como se pode ver na Secção 5.1.

O trabalho realizado para esta dissertação já foi utilizado em ambiente de competição, nomeadamente no RoboCup 2014, realizado na cidade João Pessoa, Brasil.

5.1 Trabalho futuro

Para trabalho futuro ficam identificadas, desde já, diversas situações que permitirão continuar este trabalho:

- Combinação dos vários modelos e abordagens de bola parada num só, criando um modelo robusto e funcional para qualquer ponto do campo;
- Continuação da afinação dos modelos de bola em movimento de forma a criar um modelo funcional;
- Integração dos modelos de bola parada e bola em movimento no *software* do projeto CAMBADA.

Bibliografia

[1] Página oficial CAMBADA. `http://robotica.ua.pt/CAMBADA`. Accessed: 2014-12-03.

[2] Página oficial RoboCup. `http://www.robocup.org`. Accessed: 2014-12-03.

[3] R. Dias, A. J. R. Neves, J. L. Azevedo, B. Cunha, J. Cunha, P. Dias, A. Domingos, L. Ferreira, P. Fonseca, N. Lau, E. Pedrosa, A. Pereira, R. Serra, J. Silva, P. Soares and A. Trifan. Cambada'2013: Team description paper. *Proceedings Robocup 2013*, 2013.

[4] Página oficial RoboCup MSL Referee Box. `http://msl-refbox.sourceforge.net`. Accessed: 2014-12-03.

[5] Robin Soetens, René van de Molengraft and Bernardo Cunha. RoboCup MSL - History, Accomplishments, Current Status and Challenges Ahead. *RoboCup Symposium Invited Paper on League Progress*, 2014.

[6] A. J. R. Neves, J. L. Azevedo, M. B. Cunha, N. Lau, A. Pereira, G. Corrente, F. Santos, D. Martins, N. Figueiredo, J. Silva, et al. Cambada'2010: Team description paper. *Proceedings Robocup 2010*, 2010.

[7] R. Dias, F. Amaral, J. L. Azevedo, R. Castro, B. Cunha, J. Cunha, P. Dias, N. Lau, C. Magalhães, A. J. R. Neves, A. Nunes, E. Pedrosa, A. Pereira, J. Santos, J. Silva, and A. Trifan. Cambada'2014: Team description paper. *Proceedings Robocup 2014*, 2014.

[8] Luis Almeida, Frederico Santos, Tullio Facchinetti, Paulo Pedreiras, Valter Silva, and L Seabra Lopes. Coordinating distributed autonomous agents with a real-time database: The cambada project. In *Computer and Information Sciences-ISCIS 2004*, volume 3280 of *Lecture Notes in Computer Science*, page 876–886. Springer, 2004.

[9] N. M. Figueiredo, A. J. R. Neves, N. Lau, A. Pereira, and G. Corrente. Control and monitoring of a robotic soccer team: The base station application. In *Progress in Artificial Intelligence*, volume 5816 of *Lecture Notes in Computer Science*, page 299–309. Springer, 2009.

[10] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[11] Bernarda Cecibel Sandoval Romo. Sistema multiagente web semântico para gestão de conteúdos educacionais. *Dissertação de mestrado - Universidade Estadual Paulista Julio de Mesquita Filho*, 2013.

[12] A. S. Rao and M. P. Georgeff. Bdi-agents: From theory to practice. *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, 1995.

[13] Wolfgang Bauer, Gary D. Westfall, Helio Dias. *Física para Universitários - Mecânica.* MCGRAW-HILL BRASIL, 2012.