FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

FEUP

Controlo Coordenado de Equipas de Robots Móveis

João Rodrigo Alvelos Ferreira

Mestrado Integrado em Engenharia Electrotécnica e de Computadores - Ramo de Automação, Especialização em Robótica

Orientador: António Paulo Gomes Mendes Moreira (PhD)

Co-orientador: Fernando Arménio da Costa Castro e Fontes (PhD)

Julho de 2010

Resumo

Situações reais como é o caso de determinadas aplicações militares, de busca e salvamento, de vigilância e prevenção, ou mesmo o futebol robótico, requerem o controlo coordenado de equipas de múltiplos robots móveis, muitas vezes heterogéneas e com os diversos robots a operar em meios diferentes. A necessidade de um controlo eficiente e preciso de formações de robots móveis despontou nos últimos anos um elevado interesse na comunidade científica, resultando na produção de estratégias cada vez mais complexas para coordenação de equipas dos mesmos. No entanto, a maioria da pesquisa realizada centra-se em criar e manter uma formação geométrica rígida, sem grandes preocupações em optimizar as formações tendo em vista determinado objectivos.

Propõe-se com esta dissertação o desenvolvimento de controladores de formações de robots móveis com vista a optimizar um determinado parâmetro (como será o exemplo da estimação da posição de um alvo tendo em conta dados de vários robots, ou o seguimento eficaz de um alvo por uma formação). No entanto, o controlo eficiente de uma formação não pode ser atingido sem o controlo preciso de cada um dos robots. O trabalho decorrerá portanto em duas fases: desenvolvimento de um controlador de trajectória indiviual para cada robot, e desenvolvimento de um sistema de controlo de formações perfeitamente distribuído tomando como base o controlador de trajectória já implementado. Ambos serão desenvolvidos com recurso a técnicas de controlo preditivo.

Os algoritmos produzidos serão implementados e testados na equipa de futebol robótico da FEUP (5DPO, mais precisamente nos robots da liga média). Serão apresentados resultados tanto de testes em simulação como dos robots reais, e conclusões serão tiradas a respeito da viabilidade dos controladores projectados.

Abstract

There are several real word situations such as certain military applications, search and rescue, surveillance and prevention, or even robotic soccer, that require the coordinated control of mobile robot teams. These teams are often heterogeneous and in which the various robots operate in different geographical conditions. The need for a precise and efficient control of mobile robot formations has led to an increasing interest in the topic from the scientific community in recent years, which resulted in the development of increasingly complex strategies for the coordination of robot teams. However, most of the research solely concerns the creation and stabilization of a certain geometrical formation, and not much thought is given to the optimization of robot formations having in mind a certain purpose.

In this dissertation it is proposed to design mobile robot formation controllers with the final goal of optimizing a certain parameter (such as the position estimate of a target using the data from various robots or the tracking of a certain target by a team of mobile robots traveling in formation). However, precise formation control isn't possible unless each robot is tightly controlled itself. Thus, the work on this matter will occur in two seperate phases: first the design and implementation of an individual trajectory tracking controller for each robot, followed by the development of a system for managing the robot formation, based on the trajectory tracking controller. Both will be designed using Model Predictive Control techniques.

The produced algorithms will be implemented and tested on the robotic soccer team from the Faculty of Engineering of the University of Porto (5DPO, medium league). Both simulation and real-robot tests will be performed, and conclusions shall be drawn from the results obtained as to the viability of the designed strategies.

Agradecimentos

Em primeiro lugar aos meus pais, que sempre me apoiaram e, sendo excelentes modelos de carácter, me garantiram todas as condições para que tivesse sucesso como pessoa e estudante.

Ao meu orientador, Professor António Paulo Mendes Moreira, e co-orientador Professor Fernando Arménio da Costa Castro e Fontes pela orientação e interesse demonstrado no decurso desta dissertação.

Ao Professor António Amaral, meu professor de matemática durante o ensino básico e secundário que, por sempre ensinar valorizando o raciocínio face à mecanização, foi talvez a pessoa mais influente na minha formação científica e intelectual.

À minha namorada Margarida, amigos, e restante família por serem quem são e sempre estarem lá para o que der e vier. Aos colegas de laboratório que também eles se vieram a revelar bons amigos. Aos ilustres membros do COYOTE por todas a noites longas. Aos excelentes amigos dos quatro cantos da Europa que conheci em ERASMUS, por tudo aquilo que passámos e aprendemos juntos.

A todos muito obrigado,

João Rodrigo Alvelos Ferreira

Nada mais há na vida do que beber até ao fim o vinho da iluminação e renascer outra vez. Riqueza ou miséria, ciência, glória, vexame, e a política e até a arte para tantos artistas, conhecimento do homem no corpo e no espírito – quantos modos de esquecer ou de não saber ainda o pequeno problema fundamental.

Vergílio Ferreira

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

FEUP	*Faculdade de Engenharia da Universidade do Porto*
DEEC	*Departamento de Engenharia Electrotécnica e de Computadores*
MPC	*Model Predictive Control*
NMPC	*Non-Linear Model Predictive Control*
RHCP	*Receding Horizon Predictive Control*
RPROP	*Resilient Propagation*
FCT	*Fundação para a Ciência e Tecnologia*
FPC	*Free Pascal*

Capítulo 1

Introdução

Neste capítulo inicial é dada uma introdução generalista ao projecto, enquadrando-o no âmbito de um projecto mais vasto e identificando-se a motivação por detrás do seu desenvolvimento. São ainda definidos os objectivos que se propõe atingir e descritas as fases em que decorreu o projecto.

1.1 Enquadramento e Motivação

O controlo coordenado de equipas de robots móveis, onde a manutenção de formações dos mesmos tem um papel preponderante, é uma área de estudo que tem suscitado nos últimos anos um grande interesse por parte da comunidade científica. As aplicações em que se empregam equipas de robots móveis ao invés de um só agente são cada vez mais comuns, abrangendo desde aplicações militares ou de vigilância, passando por missões de busca e salvamento, prevenção de incêndios, limpeza, ou mesmo, e porque não, o futebol robótico. Todas estas são tarefas que podem requerer ou beneficiar do controlo coordenado de equipas de robots, possivelmente heterogéneos e a operar em meios de características diferentes. O uso em aplicações como as atrás referidas de equipas de robots permite maximizar a área de operação, melhorar a percepção do meio através de partilha de informações e fusão dos dados dos sensores de cada robot, e garantir uma certa tolerância a falhas de robots individuais.

Este dissertação encontra-se incluída num projecto mais vasto financiado pela Fundação para a Ciência e Tecnologia (FCT) intitulado "*Perception-Driven Coordinated Multi-Robot Motion Control*", com uma duração de três anos. O principal objectivo do projecto centra-se no desenvolvimento de métodos de controlo de formações de veículos que permitam a alteração dinâmica da geometria da formação de modo a optimizar a exactidão da percepção cooperativa de um objecto estático ou dinâmico. Operações de seguimento de alvos ou determinação da sua velocidade e posição podem ser em muito beneficiadas com a utilização e fusão de dados dos sensores de vários robots. Interessa portanto optimizar a formação de maneira que os dados obtidos por cada

robot tenham as características ideias para realizar esta fusão. Pretende-se com esta dissertação lançar as bases para um sistema de controlo de formações que satisfaça esses requisitos.

Tanto quanto foi possível determinar, a maior parte da pesquisa e desenvolvimento já realizados nesta área foca-se na coordenação motora de múltiplos robots de modo a formar e estabilizar uma formação geométrica pretendida, como aliás será demonstrado no Capítulo 2. Esta é usualmente uma formação rígida e pré-determinada, não se encontrando na comunidade académica ou industrial muitos resultados que digam respeito a formações dinâmicas com o objectivo de optimizar um determinado parâmetro. No entanto, muitos dos conceitos utilizados nestes casos são comuns ao género de controlo que se pretende, pelo que poderão ser reaproveitados e adaptados conforme necessário.

Esta dissertação foi realizada no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores - Ramo de Automação, especialização em Robótica.

1.2 Fases do Projecto

Para que uma formação de robots móveis possa ser mantida com sucesso é primeiro necessário que cada robot empregue um controlador tão rápido e preciso quanto o possível. Ora se o interesse no controlo de formações de robots pode ser considerado relativamente recente, o controlo individual de veículos robóticos tem sido objecto de estudo recorrente ao longo de largas décadas (sendo alguns dos avanços recentes com controladores de elevado desempenho discutidos no Capítulo 2). A crescente dinâmica das aplicações em que os robots móveis são utilizados (e.g. aplicações industriais que requerem elevada precisão ou o futebol robótico que está repleto movimentos rápidos e com variações abruptas de direcção e sentido) elevaram a fasquia para o desempenho requerido dos controladores de trajectória, levando ao desenvolvimento de controladores capazes de seguir com precisão trajectórias "difíceis"a elevadas velocidades. Assim, o desenvolvimento do sistema de controlo de formações foi precedido do desenvolvimento de um controlador de trajectória de alto desempenho que irá ser empregue em cada um dos robots.

Após o desenvolvimento, optimização, e teste exaustivo do controlador de trajectória iniciou-se o desenvolvimento do controlador de formações. Este toma como base o controlador de trajectória para criar um controlador de formação perfeitamente distribuído que possa ser utilizado para gerir formações de equipas de veículos móveis, cumprindo os requisitos referidos na secção anterior. A qualidade do controlo da formação dependerá obviamente do desempenho do controlador de trajectória projectado e implementado anteriormente.

Serão utilizadas técnicas de controlo preditivo (MPC - *Model Predictive Control*) para desenvolvimento dos controladores, dado que as características deste tipo de controlo (nomeadamente ter intrinsecamente em conta as limitações e não linearidades dos sistemas reais e utilizar conhecimento futuro das trajectórias para optimizar os sinais de controlo - ver [4]) o torna o candidato perfeito para desenvolvimento de controladores de alto desempenho. Uma introdução ao controlo preditivo é dada no Capítulo 5 desta dissertação.

Como *test bed* para os algoritmos produzidos serão utilizados os robots da equipa de futebol robótico da FEUP, 5DPO - Liga Média. Serão realizados essencialmente testes em simulação (recorrendo ao software desenvolvido na faculdade de Engenharia pelo Professor Paulo Costa, SimTwo [5]), mas também serão apresentados alguns resultados obtidos com os robots reais. Todos os algoritmos de controlo serão implementados sobre o software já desenvolvido e extensamente testado para a equipa, nomeadamente a aplicação mDec, para controlo individual de cada robot, e Coach, para controlo e gestão da equipa completa. Uma descrição da equipa (nomeadamente do software utilizado) será feita no Capítulo 3.

1.3 Objectivos propostos

Os objectivos desta dissertação centram-se essencialmente no projecto e implementação de algoritmos de controlo para seguimento de trajectórias e controlo de formações baseados em controlo preditivo. Os controladores de trajectória projectados devem ser capazes de controlar eficientemente os robots em situações de velocidade elevada e trajectórias contendo mudanças bruscas de direcção e/ou de orientação. Nos controladores de formação preza-se a precisão e a estabilidade da formação, assim como a capacidade de a . Assim, e sistematizando:

- Projecto e implementação em Lazarus/FPC de um algoritmo para seguimento de trajectórias baseado em *Model Predictive Control*.
- Projecto e implementação em Lazarus/FPC de um algoritmo ou sistema para controlo coordenado da equipa de robots e gestão das formações de modo a optimizar um determinado parâmetro, também baseado em *Model Predictive Control*.
- Teste e afinação dos algoritmos desenvolvidos em ambiente de simulação e com os robots reais.
- Aplicação dos algoritmos desenvolvidos na equipa de futebol robótico 5DPO, liga média.

1.4 Estrutura do Documento

O presente documento é composto por 8 capítulos. Pretende-se detalhar com precisão o trabalho realizado, permitindo-se e facilitando-se a utilização futura por terceiros dos algoritmos desenvolvidos e implementados. Assim, no Capítulo 1 dá-se uma introdução ao projecto, expondo a problemática em causa e definindo os objectivos que se propõem atingir. No Capítulo 2 é apresentado o Estado da Arte, uma breve revisão das estratégias mais utilizadas no controlo de trajectórias e formações, assim como outras informações relevantes. Os robots utilizados são modelados dinâmica e cinematicamente no Capítulo 4, e a equipa de futebol robótico 5DPO que vai servir como plataforma de teste é descrita no Capítulo 3, assim como o software que nela é utilizado. Uma breve introdução ao *Model Predictive Control* é dada no Capítulo 5, expondo-se as vantagens e desvantagens deste tipo de controlo, descrevendo-se a teoria em que é baseado, e

justificando-se a sua escolha. O Capítulo 6 descreve o controlador de trajectória desenvolvido, e o Capítulo 7 as estratégias e algoritmos utilizados no controlo de formações. Finalmente, as conclusões a que se chegou são sistematizadas no Capítulo 8.

Capítulo 2

Estudo do Estado da Arte

Este capítulo pretende sumarizar as estratégias recentemente desenvolvidas ou habitualmente empregues no controlo de individual de robots móveis e de formações dos mesmos. Dado o quão vasta é esta área de estudo e a quantidade de pesquisa e desenvolvimento nela já efectuada (especialmente no que toca ao controlo de trajectória), será dada maior relevância aos avanços recentes na área e aos controladores mais adequados ao problema em estudo. Isto significa que o estado da arte focará essencialmente controladores de alto desempenho, capazes de operar a velocidades elevadas e com trajectórias de difícil seguimento. Não será portanto dada grande importância a soluções mais simples e que dificilmente apresentariam o desempenho desejado.

2.1 Introdução

Os dois tipos de controladores a que esta tese diz respeito (de trajectória e formação), embora directamente relacionados, são suficientemente díspares para justificar o seu estudo isoladamente. Como tal, o estado da arte de cada um deles será desenvolvido separadamente, em sub-capítulos diferentes.

2.2 Controladores de Trajectória

O papel de um controlador de trajectória é garantir o eficaz seguimento de uma trajectória de referência por parte de um robot móvel à velocidade pretendida. Esta trajectória de referência poderá ser constituída por um conjunto ordenado de pontos *(x,y)* mapeados no espaço, estando a cada ponto da trajectória associada uma orientação desejada para o robot (θ). Ao mundo está associado um referencial *(x,y)*, em relação ao qual a posição e orientação do robot são dadas. Os pontos da trajectória são também eles mapeados em *(x,y)*. Para um exemplo ver figura 2.1, que representa um referencial associado a um campo de futebol robótico e uma trajectória de referência com 3 pontos.

Muitas são as estratégias existentes para seguimento de trajectórias. Em [1], *Scolari* faz uma divisão dos métodos de controlo em duas grandes áreas: controlo reactivo, e controlo preditivo.

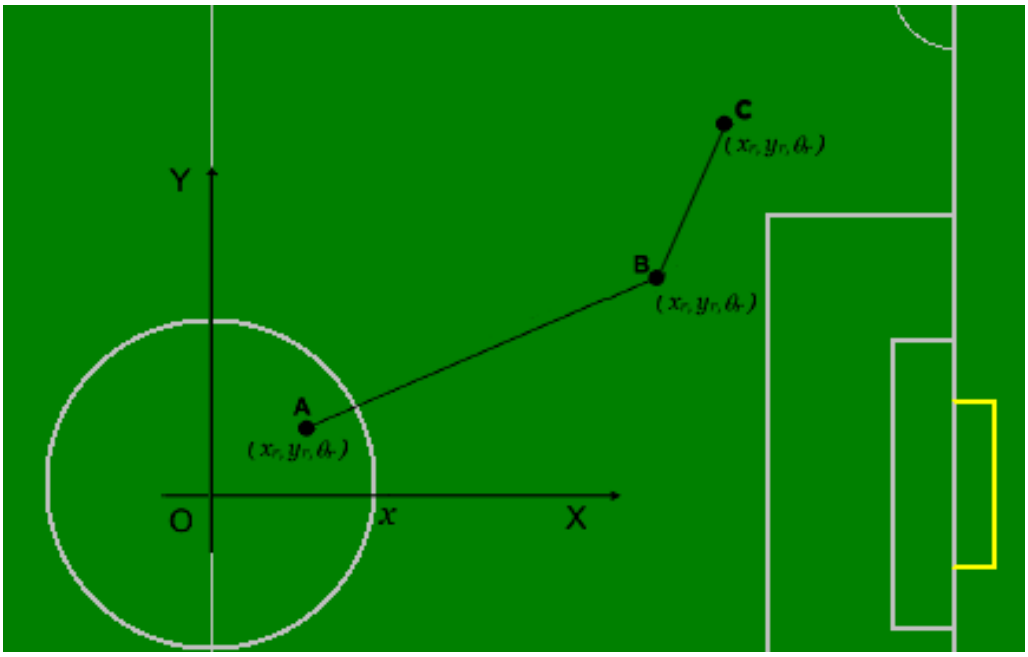


Figura 2.1: Exemplo de referencial associado ao mundo e trajectória nesse mesmo referencial

O controlo reactivo assenta numa estrutura essencialmente baseada na realimentação de estado do robot (x,y,θ) e cálculo das acções de controlo a realizar tendo em conta as diferenças entre o estado actual e a referência pretendida (para uma boa referência teórica sobre realimentação de estado, consultar [6] ou [7]). Os sinais de controlo podem ser obtidos a partir dos sinais realimentados por métodos tão diversos como *Lugar de Raízes* ou controlo *PID (Proporcional, Integral, Diferencial)* citando apenas dois deles, ambos descritos detalhadamente em [6] ou [7].

Por sua vez, o controlo preditivo é baseado em antecipação, dependendo de um modelo do processo a controlar (no caso presente, um modelo cinemático e dinâmico do robot) para prever a evolução do sistema para diferentes sinais de entrada. A entrada que é efectivamente aplicada ao robot é calculada de modo a minimizar uma determinada função custo, que penaliza essencialmente as diferenças entre a posição do robot e as posições de referência pretendidas e o esforço de controlo. A entrada calculada que minimiza a função custo é aplicada, em cada iteração, ao robot real. Simplificando largamente, o controlo preditivo pode ser basicamente resumido a um problema de optimização matemática. Uma breve introdução a esta estratégia de controlo na sua variante de *Model Predictive Control* será dada no Capítulo 5.

2.2.1 Controlo Reactivo

Estratégias relativamente eficientes e complexas de controlo reactivo de robots remontam pelo menos aos anos 80, quando em [8] *Muir e Neuman* modelaram cinematicamente um robot omnidireccional e utilizaram as equações obtidas para criar um algoritmo de controlo por realimentação. A posição do robot era determinada por odometria, por integração das velocidades medidas

nas rodas, e o controlador comparava dados das equações de cinemática e da odometria de forma a detectar perdas de tracção das rodas. Ainda dentro da realimentação simples poderão também ser citados *Keigo et al.* [9] que projectaram um controlador realimentado com malhas PI e PD, utilizando um loop de feed-forward de maneira a ter em conta nas acções de controlo os erros entre a trajectória pretendida e efectuada.

Utilizando técnicas mais actuais, *Li et al.* [10] empregaram com sucesso lógica difusa no projecto de controladores de robots móveis, implementado um controlador fuzzy para um veículo de tracção Ackerman (ver figura 2.2). O controlador recebia como entradas o erro de distância e de orientação do robot em relação à referência e fornecia como saídas a velocidade das rodas e a orientação das mesmas. Foi utilizada uma base de regras separada para o controlo da velocidade e da direcção das rodas (na realidade, existiam dois controladores diferentes), com 5 funções de pertença para cada variável de entrada.

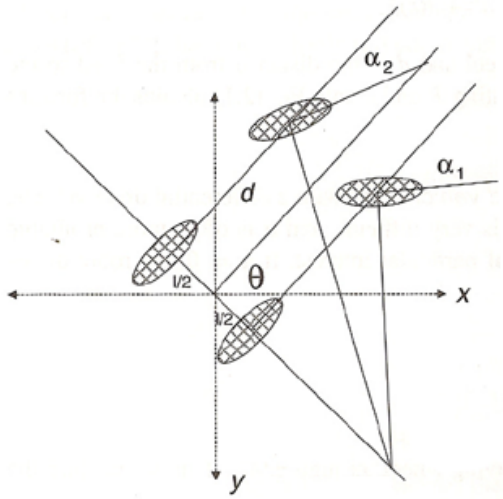


Figura 2.2: Veículo de tracção Ackerman

Mais recentemente, uma solução interessante para seguimento de trajectórias por robots omnidireccionais é proposta por *Scolari* em [1] e sintetizada por *Scolari, Moreira, e Costa* em [11]. O algoritmo é baseado na realimentação parcial do estado do robot (posição e orientação em relação ao mundo) e no facto de que uma trajectória pode ser aproximada por vários segmentos de recta. Esses segmentos de recta são por sua vez definidos por dois pontos, o que permite que para cada ponto da trajectória seja definida um velocidade e orientação diferentes. Um esquema da definição da trajectória e do estado do robot no mundo é apresentado na figura 2.3. Em cada loop do algoritmo as entradas de controlo do robot (referências de velocidade, velocidade normal, e velocidade angular) são calculadas de modo a corrigir a sua posição e orientação em relação à posição e orientação de referência para o ponto actual da trajectória. Estas velocidades de referência podem provocar a saturação de um ou mais motores do robot, pelo que é aplicado um processo de escalonamento das velocidades dos motores de modo a manter a direcção do movimento no caso de um

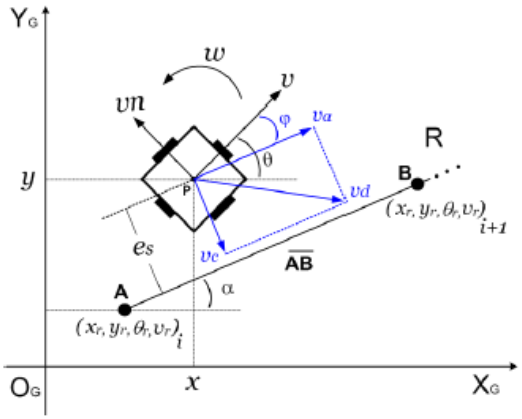


Figura 2.3: Esquema de funcionamento do controlador reactivo implementado por *Scolari* (retirado de [1])

dos motores saturar. Este algoritmo requer uma modelização precisa dos robots e encontra-se descrito em [12]. Uma vez escalonadas, se necessário, as referências são aplicadas ao robot. Este controlador encontra-se actualmente aplicado nos robots da equipa 5DPO, sendo as trajectórias geradas por uma versão modificada do algoritmo A* (ver [13]). Esta trajectória é gerada em cada iteração do loop de controlo, e é considerado como referência o segundo ponto da trajectória a partir da posição actual do robot. Isto confere ao controlador um certo grau de "predição", dado que a cada momento o robot considera como a posição desejada não o ponto seguinte da trajectória mas o terceiro ponto a partir da posição actual, o que lhe permite de certa forma antecipar mudanças bruscas de direcção. Em [14] é proposto um controlador baseado no seguimento de veículo virtual. Neste caso, o movimento do ponto de referência na trajectória é definido por uma equação diferencial, sendo assumido que o controlo dos robots para seguimento desses pontos é realizado pelos habituais reguladores proporcionais. Representa um controlador de mais alto nível que os descritos anteriormente, mas ainda assim introduz a nova noção de *veículo virtual*.

Nascimento desenvolveu com bons resultados um controlador multivariável linearizante em cascata para robots omni-direccionais em [15]. A técnica utilizada consiste em controlar o sistema considerando-o linear em malha fechada. *Nascimento* obteve uma modelização multi-variável dinâmica e cinemática do robot em espaço de estados, projectando depois uma lei de controlo que conserva o acoplamento das velocidades das rodas.

Todas as estratégias aqui descritas partilham uma característica comum: as acções de controlo dependem do valor do erro actual, que consiste na diferença entre o estado actual do robot (posição e orientação em relação ao mundo) e o estado pretendido (posição e orientação de referência). Isto significa que as acções de controlo para corrigir um determinado erro só serão tomadas depois de este existir. Tomando a analogia que *Camacho e Bordons* fazem em [16], é um pouco como conduzir um carro olhando apenas para o retrovisor e não considerando a estrada em frente. O controlo preditivo pretende de certa maneira colmatar esta falha.

2.2.2 Controlo Preditivo

O crescente interesse em controlo preditivo - MPC (*Model Predictive Control*) - ao longo dos últimos anos tem levado à aplicação deste tipo de técnicas no desenvolvimento de novas soluções para controlo de robots móveis, muitas vezes aliando ao MPC técnicas como redes neuronais, lógica difusa, ou algoritmos genéticos. A natureza antecipativa do controlo preditivo torna-o uma opção extremamente interessante para controlo de seguimento de trajectórias, uma vez que tem internamente em conta tanto as características reais do robot (como a sua dinâmica, velocidades máximas, ou zona morta de motores, partindo do princípio que o modelo utilizado é suficientemente preciso) como a posição dos pontos futuros da trajectória a seguir.

Kanjanawanushkul e Zell, em [17], propõem um controlador para seguimento de trajectórias por um robot omni-direccional conjugando a formulação clássica de MPC com uma técnica de seguimento de veículo virtual. Linearizando o sistema em torno do ponto de operação, transformam o problema de optimização num problema de programação quadrática que é resolvido em cada instante de controlo, sendo a solução óptima aplicada ao robot. Um controlador MPC para um robot móvel de tracção Ackerman é apresentado em [18]. Este tem a particularidade de incorporar durante a optimização informação sobre obstáculos detectados pelos sensores, recorrendo a métodos numéricos (gradiente descendente) para minimização da função custo. Uma solução semelhante é apresentada por *Klancar et al.* em [19], que propõe um controlador preditivo baseado num modelo dinâmico linearizado e com uma função custo que penaliza tanto os erros de seguimento como o esforço de controlo (o valor das entradas de controlo). Limites de velocidade e de aceleração são incluídos no modelo de forma a prevenir perdas de tracção das rodas.

Estratégias de controlo bem mais interessantes podem ser conseguidas conjugando várias técnicas que optimizem ao máximo os vários componentes do controlador preditivo. *Gu et Al.* desenvolvem em [20] um algoritmo MPC que utiliza também o método do gradiente descentente para minimização da função custo. A principal característica que o distingue dos demais é a utilização de uma rede neuronal para modelar as não linearidades do robot (dado que as características destas as tornam ideais para modelizar sistemas complexos e aproximar funções contínuas arbitrárias). Em [21] são aplicados algoritmos genéticos para minimização em tempo real da função custo, sendo que as limitações de velocidade são também tidas em conta pelo controlador. Um outro exemplo desta mistura de técnicas encontra-se em [22]. *Jiang et Al.* apresentam um controlador que combina lógica difusa com controlo preditivo, utilizando a primeira para lidar com as não-linearidades do sistema e o segundo para prever a posição e orientação do robot.

A referência principal para o controlador de trajectória desenvolvido nesta dissertação é o controlador proposto e desenvolvido por *Scolari* em [1] e sistematizado em [23] e [24] por *Scolari et Al.*. A estrutura deste controlador está demonstrada na figura 2.4.

Este controlador toma a ideia básica da formulação clássica do MPC e implementa uma função custo que penaliza tanto as diferenças de posição e orientação do robot em relação às posições e orientações de referência como a variação das entradas de controlo (de modo a minimizar o esforço de controlo). Métodos numéricos são utilizados para minimizar a função custo, sendo feitos testes

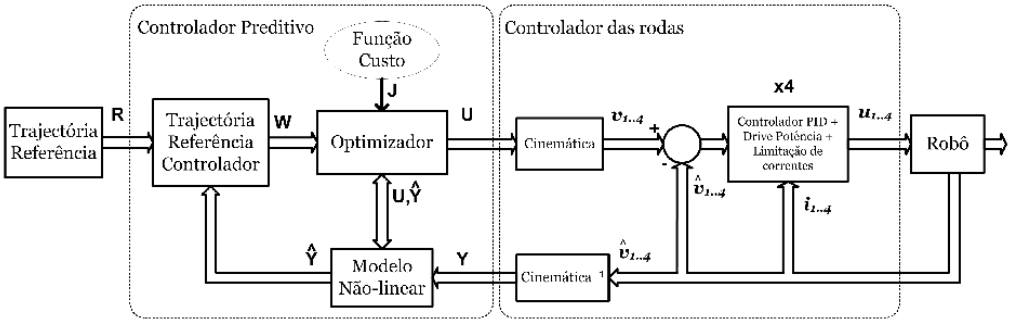


Figura 2.4: Estrutura do controlador preditivo implementado por *Scolari* (retirado de [1])

com algoritmos de gradiente descendente (*steepest descent*) e gradientes conjugados (*Fletcher-Reeves* e *Polak-Ribiere*). Para uma referência sobre estes métodos de optimização poderá ser consultado [25]. O controlador assume que em cada iteração está disponível a trajectória completa que o robot deve seguir e calcula a partir desta a trajectória de referência para os próximos N instantes, tendo em conta a posição actual do robot e o módulo da velocidade pretendida (ver figura 2.5).

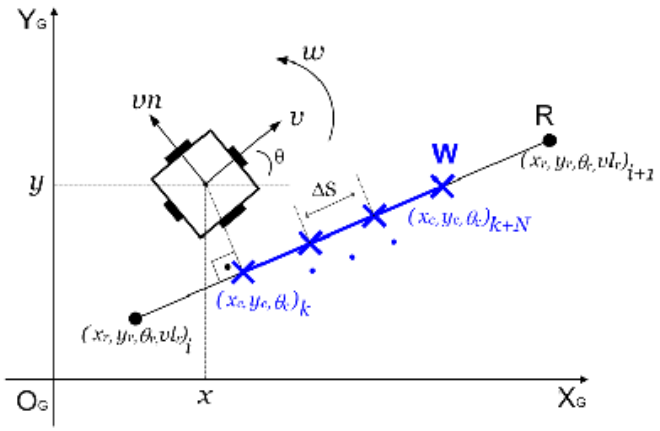


Figura 2.5: Trajectória de referência para o controlador (retirado de [1])

O modelo do robot é utilizado em conjunto com esta trajectória de referência para prever a evolução do sistema para várias entradas, e a função custo é minimizada utilizando estes dados através de um dos métodos de gradiente referidos acima. A grande vantagem deste controlador consiste no modelo extremamente completo que foi desenvolvido para o robot. Todos os elementos que influenciam o movimento estão modelizados nas equações diferenciais que os regem (motores, controladores dos motores, atritos, dinâmica de rotação e dinâmica de translação do robot), e todos os parâmetros destas equações foram afinados para corresponderem muito precisamente aos do robot real. Isto resulta numa simulação extremamente precisa da evolução do estado do robot, o

que se traduz na obtenção de sinais de controlo o mais adequados possíveis à trajectória a seguir e consequentemente, num excelente desempenho do controlador. Para mais, os métodos numéricos utilizados para minimizar o valor da função custo são suficientemente leves computacionalmente para poderem ser utilizados em tempo real tendo em conta o período de controlo em causa (*40ms*).

As referências citadas correspondem a algumas das soluções mais interessantes encontradas para controlo de seguimento de trajectórias recorrendo a controlo preditivo. A grande vantagem do MPC em relação ao controlo reactivo é óbvia e pode ser resumida a uma palavra: antecipação.

2.3 Controladores de Formação

Os campos da coordenação e controlo de formação de robots móveis têm sido alvo de estudo continuado nos últimos anos, tendo a coordenação de equipas de múltiplos robots vindo a ocupar um papel de destaque no mundo da robótica móvel. São várias as definições existentes para o problema do controlo de formações. Em [26] este problema é descrito como "seguimento coordenado de uma trajectória onde múltiplos veículos seguem uma trajectória pré-especificada enquanto mantêm uma formação fixa entre veículos". As vantagens da utilização de equipas de múltiplos robots incluem a robustez, flexibilidade, e a adaptabilidade a ambientes dinâmicos desconhecidos ([3]). Estas são claramente importantes quando se considera aplicações como missões de busca e salvamento, mapeamento de fundos oceânicos, detecção de fogos florestais, detecção e remoção de minas, ou até futebol robótico.

Em [27] é feita uma divisão das estratégias de controlo de formações em três grandes grupos: seguimento de líder, estrutura virtual, e *behavior-based*. Existem também aproximações puramente baseadas em controlo preditivo, onde cada robot da formação tem um papel idêntico. A aproximação por seguimento de líder é aparentemente a mais utilizada, pela quantidade imensa de artigos que a referem. Baseia-se na existência de um robot (real ou não) que segue de forma precisa a trajectória desejada enquanto que os restantes robots da formação seguem este, mantendo uma determinada distância e posicionamento relativo. A grande maioria das estratégias de controlo deste tipo emprega controladores preditivos, sendo algumas delas descritas mais abaixo. As estratégias baseadas em estrutura virtual tratam a formação de robots como um corpo rígido onde todos os pontos mantêm uma posição fixa, sujeita a constrições físicas, e onde qualquer perturbação feita a um dos pontos se propaga a todos os outros pontos da estrutura. Isto é conseguido utilizando estratégias de controlo relativamente simples, como por exemplo as descritas em [28] e [29]. Já com uma aproximação *behaviour-based* são utilizados em cada robot comportamentos reactivos para controlo da formação, sendo integrados com outros comportamentos para seguimento de trajectórias e detecção e desvio de obstáculos. Um bom exemplo de aplicação desta técnica encontra-se em [30]. De seguida descrevem-se em maior pormenor algumas das soluções mais pertinentes encontradas.

Uma aplicação bastante interessante baseada em seguimento de líder e utilizando controlo preditivo é apresentada por *Zell* e *Kanjanawanishkull* em [27]. Nesta solução o problema é dividido em dois sub-problemas essencialmente desacoplados: o do robot líder e o dos robots

seguidores. O robot líder segue uma trajectória fixa utilizando uma estratégia semelhante a [17], transmitindo a sua velocidade, estado, e trajectória prevista aos restantes robots da formação. Os robots seguidores utilizam esta informação em conjunto com a definição da formação desejada para estimar a sua própria trajectória de referência. Esta é depois seguida utilizando o mesmo controlador que o robot líder emprega. O problema de gestão da formação é descentralizado, dividindo o sistema completo em múltiplos sub-sistemas que são independentemente controlados, o que permite horizontes de predição e controlo maiores. Cada robot resolve o seu próprio problema de optimização para seguir a sua trajectória estimada.

Os mesmo autores apresentam em [31] uma outra estratégia para controlo de formações baseada em MPC, onde cada robot deve seguir a sua própria trajectória de referência enquanto mantém a formação desejada com os restantes robots do grupo. É assumido que cada robot tem uma trajectória de referência pré-especificada e que o que o controlador determina essencialmente são as velocidades com que cada robot a segue. Neste caso a interacção entre os robots da formação é conseguida acoplando as funções custo dos controladores de cada agente. De um modo geral cada robot resolve localmente a solução do seu problema de controlo e comunica a informação mais recente aos seus vizinhos. Estes por sua vez resolvem o seu próprio problema de minimização tendo em conta estes dados na sua função custo. São ainda empregues técnicas de *morphing* para gerar formações intermédias no caso de troca de uma formação para outra em movimento.

Em [2] *Fontes et Al.* propõem um controlador preditivo de duas camadas para executar o controlo de formações de veículos móveis. Consideram que existem dois sub-problemas a resolver de modo a cumprir este objectivo: o controlo de trajectória (determinar uma trajectória e correspondentes sinais dos actuadores para a formação como um todo) e o controlo de formação (alterar os sinais de controlo dos actuadores em cada veículo de modo a compensar pequenas variações em torno da trajectória de referência e assim manter a posição relativa entre veículos). Uma vez que são considerados veículos não holonómicos, os dois problemas são intrinsecamente diferentes. Quando se trata de um único veículo a em movimento para seguir uma trajectória fixa este não se pode movimentar em todas as direcções (veja-se a manobra de estacionamento da figura 2.6), pelo que é necessário um controlador não linear que permita leis de controlo e feedback descontínuas.

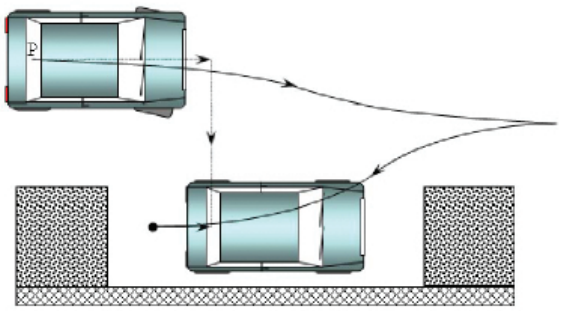


Figura 2.6: Manobra de estacionamento de veículo não holonómico (retirado de [2])

Já no caso de um veículo a mover-se como parte de uma formação (por exemplo, na manobra de ultrapassagem da figura 2.7), a posição relativa entre os veículos pode ser modificada em qualquer direcção, como se fossem holonómicos. Para lidar com este problema, um controlador linear é apropriado.

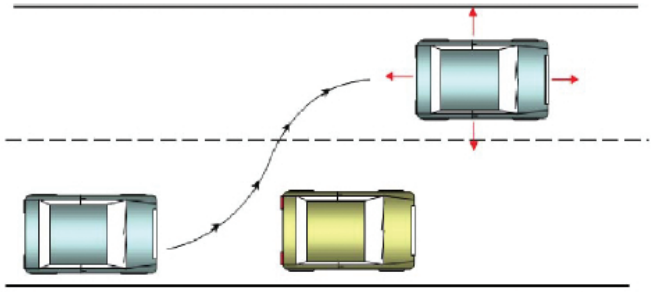


Figura 2.7: Manobra de ultrapassagem de veículo não holonómico (retirado de [2])

A arquitectura de duas camadas é adaptada a esta estrutura: um controlador preditivo não-linear do tipo *bang-bang* é utilizado para computar de forma centralizada uma lei de controlo para seguimento de trajectória, sendo um pequeno conjunto de parâmetros transmitidos a cada robot em cada iteração. Localmente, um controlador preditivo linear é utilizado para compensar as pequenas alterações em torno da trajectória de referência de maneira a manter a formação desejada. As limitações do sistema são explicitamente impostas no controlador MPC. Esta implementação tem a particularidade de que nenhum problema de optimização tem de ser resolvido em tempo real, resultando num algoritmo simples e eficiente.

Lim et Al. descrevem também uma estratégia de controlo por seguimento de líder utilizando controladores NMPC distribuídos ([3]), com a diferença que este líder não segue uma trajectória fixa mas é controlado manualmente por um agente humano. Neste caso a referência futura da posição do líder não está disponível, pelo que existe uma dificuldade acrescida na implementação da formação. No entanto, a posição e velocidade actuais do líder são transmitidas aos restantes robots da formação que utilizam estes dados para estimar as trajectórias futuras a seguir de modo a manter a formação. Os limites do sistema são implementados adicionando uma função de penalização à função custo. A formação propriamente dita é definida, para cada robot, pela distância ao robot líder e ao ângulo em relação ao sistema de coordenadas definido por este (ver imagem 2.8).

Muitas outras estratégias de controlo de formações foram propostas até à data. Em [32] é primeiro desenvolvido um algoritmo de seguimento de trajectórias para cada robot utilizando técnicas baseadas em Lyapunov. As trajectórias de referência para cada um dos agentes da formação são obtidas por translação da trajectória de referência do líder ou utilizando conjuntos de circunferências com o mesmo centro e diferentes raios. A formação é mantida controlando as velocidades de cada robot ao longo da sua própria trajectória de referência. Já em [33] não existe uma formulação líder/seguidor, ocupando todos os veículos o mesmo papel na formação. Esta é mantida com recurso a um controlador preditivo não linear que gere o controlo do sistema completo.

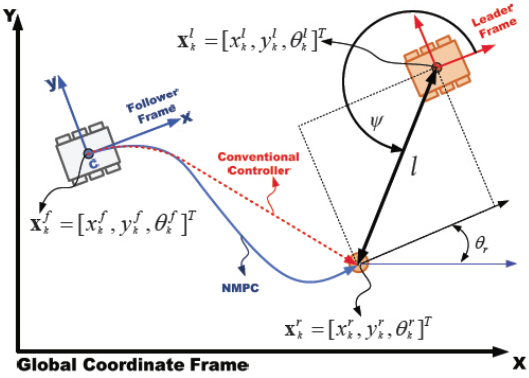


Figura 2.8: Veículo líder e seguidor (retirado de [3])

As estratégias atrás descritas para controlo de formações são uma selecção das soluções mais pertinentes encontradas durante a recolha de dados para estudo do estado da arte. Pretende-se que a sua referência dê ao leitor uma ideia geral do estado actual desta matéria, assim como que introduza algumas das ideias que são utilizadas e desenvolvidas no projecto do controlador de formações desta dissertação.

2.4 Conclusão

No que toca aos controladores para seguimento de trajectória foi efectuada uma breve revisão de algumas das estratégias de alto desempenho utilizadas actualmente no controlo de robots móveis. Foi feita uma grande divisão destes controladores em controladores preditivos e reactivos, sendo expostos alguns exemplos pertinentes de cada tipo. Do estudo dos dois tipos de algoritmos de controlo pode-se sumarizar que a grande vantagem dos controladores preditivos em relação aos reactivos consiste na possibilidade de utilizar informação futura do sistema (sendo a sua evolução prevista através de um modelo do mesmo e das referência pretendidas) para calcular os sinais de controlo, o que permite antecipar situações potencialmente problemáticas como mudanças bruscas de direcção, acelerações e desacelerações bruscas, e não linearidades no comportamento do sistema. Têm no entanto a grande desvantagem de requerer métodos de optimização matemática que podem não ser viáveis de implementar para controlo em tempo real e de necessitar de um modelo preciso do sistema a controlar. Estes factos tornam a sua implementação mais complexa que a dos controladores reactivos. Para mais, sendo uma área relativamente recente, ainda existem algumas dúvidas quanto às garantias de estabilidade que estes controladores oferecem. Ainda assim, e tendo em conta tudo o que foi escrito, é claro o potencial que este tipo de controlador oferece para o projecto de controladores de trajectória eficientes.

Relativamente aos controladores de formação foram referidas algumas das soluções mais recentes e/ou mais pertinentes. A divisão das estratégias de controlo em três grandes grupos (seguimento de líder, estrutura virtual, e *behavior-based*) não é tão exclusiva como aquela feita para

os controladores de trajectória, uma vez que foram apresentados controladores que simplesmente não encaixam em nenhuma destas categorias. Pela sua simplicidade, a utilização de controlo por seguimento de líder parece ser a mais comum, surgindo em diversas formas em muitas das soluções encontradas. Aliado a esta estratégia, surge muitas vezes o controlo preditivo, com todas as vantages e desvantagens que lhe estão associadas e foram discutidas no parágrafo anterior.

Espera-se que este capítulo reflicta uma imagem clara do estado actual do controlo de seguimento de trajectórias e formações, introduzindo diversas ideias e estratégias de controlo que serão utilizadas no desenvolvimento dos controladores que são o objecto de estudo desta dissertação.

Capítulo 3

A Equipa de Futebol Robótico 5DPO

Este capítulo pretende descrever brevemente a equipa de futebol robótico 5DPO que foi utilizada como plataforma de testes para os algoritmos desenvolvidos no decurso desta tese. Assim, serão sucintamente descritos tanto os robots pertencentes à equipa como a infra-estrutura de software utilizada no seu controlo.

3.1 Introdução

A equipa de Futebol Robótico 5DPO foi criada no final da década de 90 na Faculdade de Engenharia da Universidade do Porto para competir na RoboCup [34]. Tem desde então sofrido considerável evolução, com alterações e melhoramentos a serem introduzidos todos os anos.

Os robots da Liga Média 5DPO foram utilizados como plataforma de teste para os controladores desenvolvidos nesta dissertação. Ainda mais importante, os algoritmos foram implementados sobre a infra-estrutura de software então utilizada para controlar a equipa, pelo que a sua descrição é importante para a compreensão do trabalho realizado neste projecto.

3.2 Robots

A equipa é constituída por cinco robots idênticos, onde quatro tomam a posição de jogadores de campo e um quinto o lugar do guarda-redes. Estes são robots omni-direccionais de três rodas (cuja cinemática será descrita no capítulo 4), com uma altura de 80 *cm* e um diâmetro máximo de cerca de 50 *cm*. Um dos robots pode ser visto na figura 3.1.

Cada robot possui um computador portátil que corre os algoritmos de controlo, fazendo todos os robots parte de uma rede local que lhes permite trocar informações com a máquina central e receber ordens da mesma.

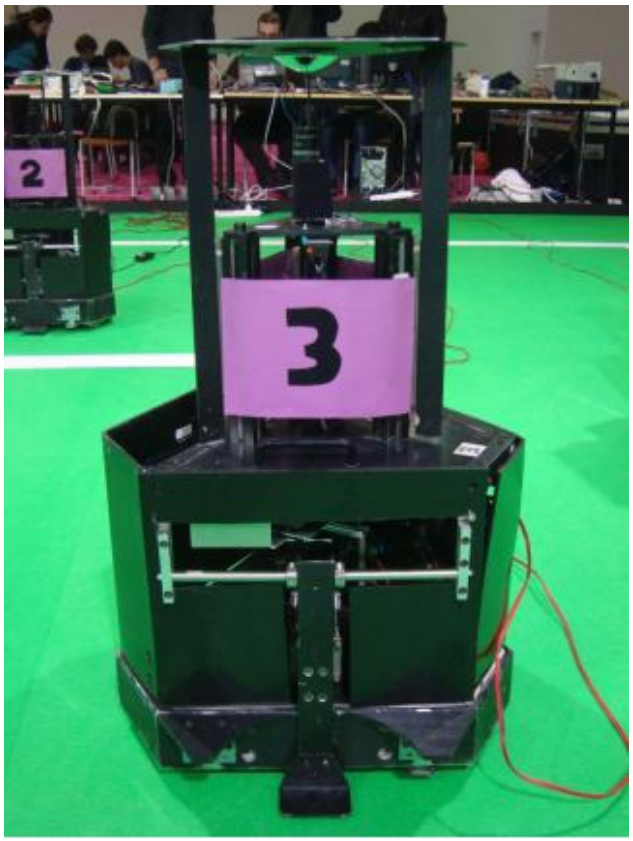


Figura 3.1: Um dos robots da equipa 5DPO

3.2.1 Motorização

Cada roda omni-direccional está acoplada a um motor DC controlado por um driver desenvolvido na FEUP. Este possui uma malha PI para controlo de velocidade e recebe as referências de velocidade do computador portátil através de comunicação série (variante RS232). As portas série de cada driver estão ligadas em daisy-chain, pelo que apenas uma porta série é necessária no portátil.

3.2.2 Localização

Cada robot localiza-se individualmente utilizando um misto de odometria e de um sistema de visão artificial omni-direccional. O algoritmo de localização baseado na visão omni-direccional, desenvolvido por *Gouveia* em [35], assenta na correspondência de um mapa pré-determinado ao local observado pelo robot a cada momento. A imagem de 360 graus é obtida com recurso a um espelho côncavo ao qual a câmara está apontada e com o qual se encontra perfeitamente alinhada. Uma vez processada a imagem tendo em conta a curvatura do espelho são identificadas as linhas do campo através da cor e é feita uma correspondência da imagem observada ao mapa conhecido

do campo. Assim, a localização mais provável do robot é obtida a cada momento e utilizada para corrigir os dados da odometria.

3.3 Software

O controlo de cada robot individualmente e da equipa como um todo é realizado por um conjunto de três programas (OVIS, mDec, e Coach) em permanente comunicação via protocolo UDP. Todas as máquinas envolvidas fazem parte de uma mesma rede local wireless, configurada de propósito para este efeito. A cada robot está associado um IP fixo dentro da gama da rede utilizada, assim como à máquina central.

3.3.1 Arquitectura de Controlo

Os três programas - OVIS, mDEC, e Coach - operam em conjunto de modo a garantir o funcionamento da equipa. Um esquema das suas interacções pode ser observado na figura 3.2. A comunicação entre aplicações, mesmo no caso em que estejam a correr no mesmo computador (o que é comum para o mDec e o OVIS) ocorre sempre em UDP. Os algoritmos desenvolvidos neste projecto baseiam-se nesta arquitectura, sendo os controladores dos robots implementados sobre a *framework* fornecida pelo mDec. O Coach também foi modificado de modo a melhor servir os propósitos pretendidos.

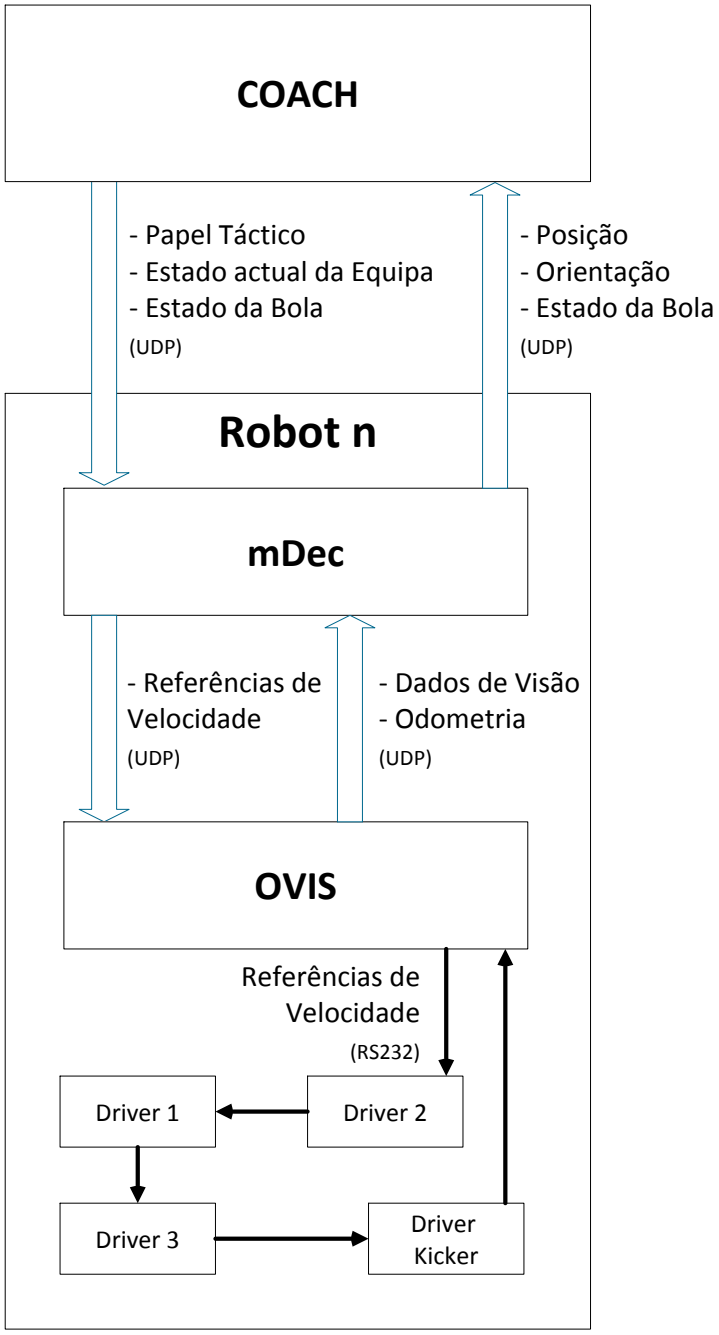


Figura 3.2: Arquitectura de software da equipa 5DPO

OVIS - Faz a ponte entre o hardware e o software. Cada robot corre uma instância do OVIS no seu computador, sendo este responsável por adquirir e processar os dados da câmara e da odometria enviando-os de seguida ao mDec correspondente. O OVIS comunica também com os drivers dos motores e do kicker através do protocolo série RS232, enviando-lhes as referências de velocidade de cada motor e a ordem de disparo do kicker. O seu interface pode ser visto na figura 3.3.

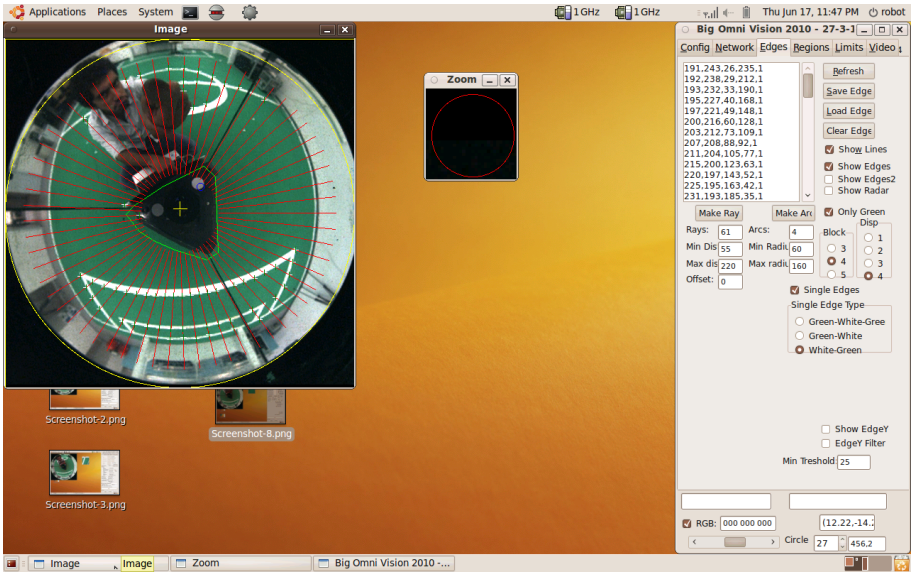


Figura 3.3: Screenshot do OVIS

mDec - É o software de controlo de cada robot (figura 3.4). Recebe do OVIS os dados da posição e da odometria, processando-os de modo a localizar o robot no espaço. De acordo com o seu papel táctico actual corre os controladores necessários para calcular as referências de velocidade para os motores, enviando-as de seguida ao OVIS correspondente por UDP. Porque comunica com o OVIS por UDP não tem necessariamente que estar a correr no portátil do robot que está a controlar, podendo ser executado remotamente. Os algoritmos de controlo aqui desenvolvidos são implementados sobre o mDec.

Coach - O "treinador"da equipa (figura 3.5). Recebe de cada mDec a posição e orientação do robot correspondente e a posição da bola vista por esse robot. Constrói assim uma imagem do estado actual da equipa, que vai enviar a cada robot. Dependendo das posições dos robots, da bola, e do estado de jogo actual, determina a táctica a utilizar e envia a cada mDec o papel táctico que determinou para ele. Envia também a posição da bola mais provável, calculada a partir da posição reportada por cada robot. Uma única instância do Coach é executada, numa máquina central em comunicação com as máquinas que correm os mDecs de cada robot.

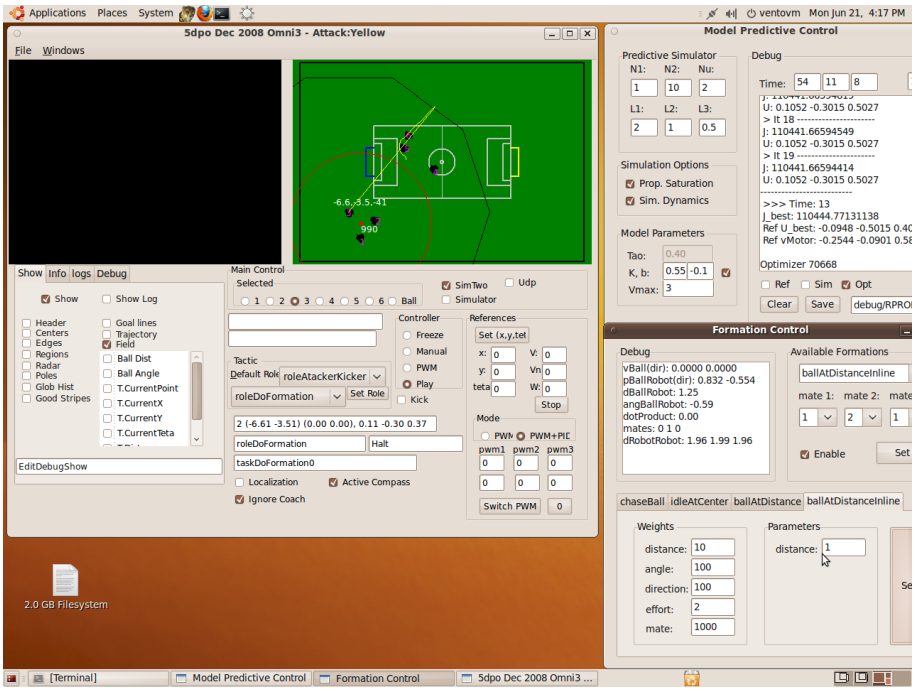


Figura 3.4: Screenshot do mDec

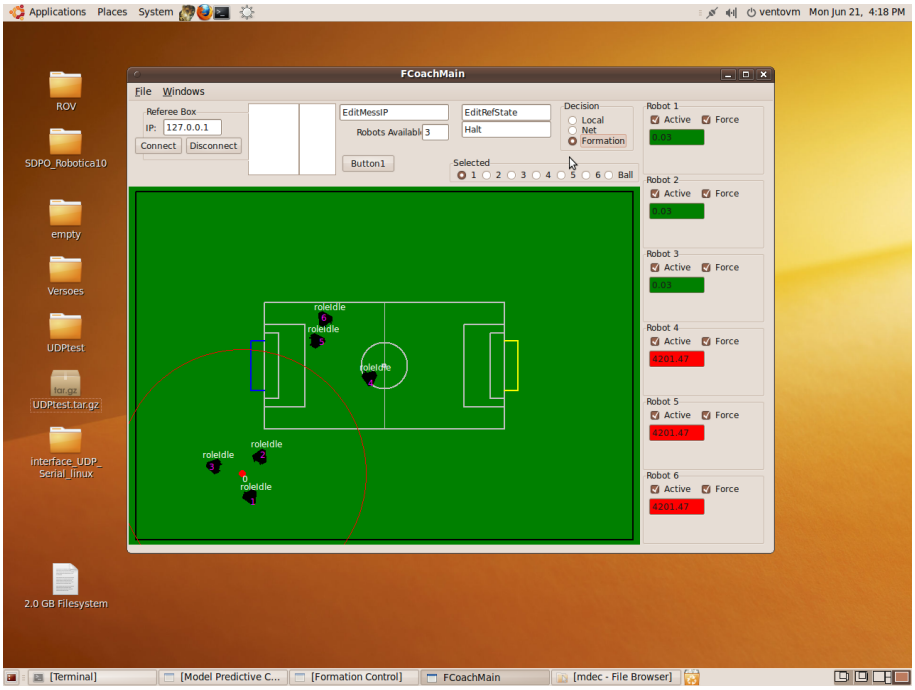


Figura 3.5: Screenshot do Coach

3.4 Conclusão

A arquitectura de software utilizada na equipa pode ser adaptada com relativa facilidade para servir os propósitos de controlo de trajectórias e formações que são o objectivo principal desta dissertação. Em particular, o mDec fornece já dados sobre a posição e orientação actuais do robot e as funcionalidade necessárias para enviar referências de velocidade para o hardware. A utilização de um sistema já extensivamente testado é uma mais-valia e permite concentrar esforços no desenvolvimento dos algoritmos de controlo ao invés dos problemas de localização ou de interface com o hardware. No entanto, também significa que os controladores aqui desenvolvidos, quando executados no robot real, vão estar sujeitos às limitações do sistema de localização já implementado e dos controladores de velocidade utilizados nos motores.

Capítulo 4

Modelação dos Robots Utilizados

O capítulo actual apresenta o modelo cinemático e dinâmico dos robots utilizados como plataforma de teste, essencial para a compreensão dos controladores projectados.

4.1 Introdução

Os algoritmos de controlo desenvolvidos nesta dissertação foram projectados para os robots da liga média da equipa de futebol robótico da Faculdade de Engenharia da Universidade do Porto, 5DPO. Este capítulo apresenta os modelos cinemático (incluindo as equações de cinemática inversa e directa) e dinâmico destes robots. Este assunto foi previamente estudado em [36], para robots omni-direccionais de três e quatro rodas. Em [1] e [37] são apresentados métodos para estimação rigorosa dos parâmetros do modelo dinâmico de um robot omni-direccional de quatro rodas.

O modelo do robot em causa encontra-se esquematizado na figura 4.1.

- x_c, y_c $[m]$ - Posição do centro de massa do robot, em coordenadas do sistema global.
- θ $[rad]$ - Orientação do robot em relação ao eixo x do sistema de coordenadas global.
- v, v_n $[m/s]$ - Velocidade linear do robot, nas coordenadas do sistema de coordenadas do mesmo.
- w $[rad/s]$ - Velocidade angular do robot.
- v_x, v_y $[m/s]$ - Velocidade linear do robot, em coordenadas do sistema global.
- v_1, v_2, v_3 $[m/s]$ - Velocidade linear das rodas do robot.
- w_1, w_2, w_3 $[rad/s]$ - Velocidade angular das rodas do robot.
- d $[m]$ - Distância da roda ao centro de massa.

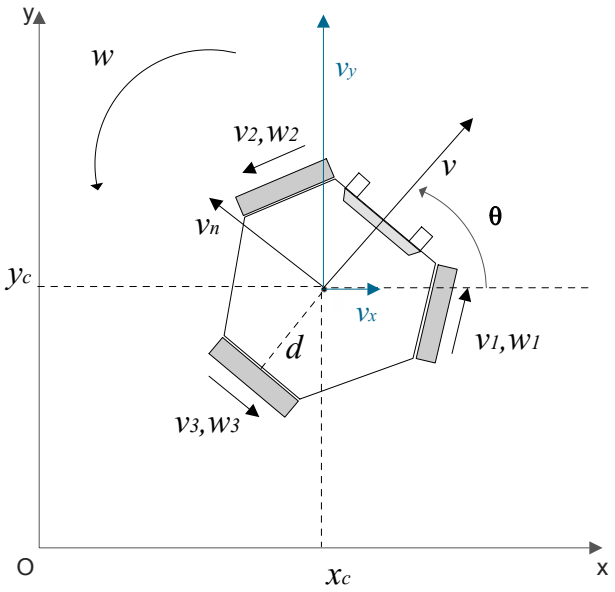


Figura 4.1: Robot omni-direccional de três rodas

Defina-se, para as velocidades do robot, os vectores V_R (velocidades do robot nas coordenadas do mesmo), V_0 (velocidades do robot nas coordenadas globais), e V_w (velocidades das rodas do robot):

$$V_R(t) = [\, v(t)\ v_n(t)\ w(t)\,]^T \qquad V_0(t) = [\, v_x(t)\ v_y(t)\ w(t)\,]^T \qquad V_w(t) = [\, v_1(t)\ v_2(t)\ v_3(t)\,]^T$$

E ainda, para o estado do robot (a sua posição e orientação), o vector X:

$$X(t) = [\, x_c(t)\ y_c(t)\ \theta(t)\,]^T$$

4.2 Parâmetros do Robot

Os parâmetros geométricos e físicos do robot real utilizados no processo de simulação são apresentados na tabela 4.2.1.

Parâmetros do robot Real	
d [m]	0.195
r [m]	0.05
h [m]	0.8
g	12:1

Tabela 4.2.1: Parâmetros físicos do Robot

Aqui, d é a distância de cada roda ao centro de massa do robot, r o raio de cada roda, h a altura total do robot, e g a redução da caixa de velocidades do motor.

4.3 Sistemas de Coordenadas

Consideram-se dois sistemas de coordenadas distintos: um associado ao mundo, e outro associado ao robot.

O sistema de coordenadas associado ao mundo, denominado sistema de coordenadas global, é definido pelos eixos XoY e é um referencial fixo relativamente ao espaço. O sistema de coordenadas do robot é definido pelo vector unitário na mesma direcção da orientação do robot e pelo vector perpendicular a este. As matrizes para transformação de coordenadas entre estes dois referenciais são dadas nas equações 4.1 e 4.2, no âmbito da transformação dos valores da velocidade entre os dois referenciais.

Tem-se, para transformar as velocidades linear e angular de coordenadas do sistema global para o sistema de eixos do robot:

$$V_R = \begin{bmatrix} cos(\theta(t)) & sin(\theta(t)) & 0 \\ -sin(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot V_0 \tag{4.1}$$

E, multiplicando ambos os membros da equação pela matriz inversa:

$$V_0 = \begin{bmatrix} cos(\theta(t)) & -sin(\theta(t)) & 0 \\ sin(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot V_R \tag{4.2}$$

4.4 Modelo Cinemático

Apresentam-se de seguida as equações que regem a cinemática inversa e directa do robot, assim como a equação de evolução da posição do mesmo.

4.4.1 Cinemática Directa e Inversa

As velocidades das rodas, tendo em conta um modelo unicamente cinemático, podem ser determinadas a partir das velocidades lineares do robot através da equação 4.3:

$$\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \begin{bmatrix} sin(\pi/3) & cos(\pi/3)) & d \\ -sin(\pi/3) & cos(\pi/3) & d \\ 0 & -1 & d \end{bmatrix} \cdot \begin{bmatrix} v(t) \\ v_n(t) \\ w(t) \end{bmatrix} \tag{4.3}$$

Aplicando cinemática inversa, as velocidades individuais das rodas estão relacionadas com as velocidades linear e angular do robot pela equação 4.4:

$$\begin{bmatrix} v(t) \\ v_n(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} \sqrt{3}/3 & -\sqrt{3}/3 & 0 \\ 1/3 & 1/3 & -2/3 \\ 1/(3d) & 1/(3d) & 1/(3d) \end{bmatrix} \cdot \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} \tag{4.4}$$

4.4.2 Evolução da Posição

Considerando o sistema em tempo discreto, sendo T o tempo de amostragem, a posição do robot no instante $k+1$ pode ser determinada a partir da sua posição e velocidades no instante k resolvendo a equação 4.5:

$$X(k+1) = X(k) + T \cdot \begin{bmatrix} cos(\theta(k)) & -sin(\theta(k)) & 0 \\ sin(\theta(k)) & cos(\theta(k)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot V_R \tag{4.5}$$

4.5 Modelo Dinâmico

As dinâmicas de translação do robot encontram-se descritas de acordo com o princípio fundamental da mecânica de Newton nas equações 4.6 e 4.7. A sua dinâmica de rotação está descrita pela equação 4.8.

$$M \cdot \frac{dv(t)}{dt} = \sum F_v(t) + F_{Bv}(t) + F_{Cv}(t) \tag{4.6}$$

$$M \cdot \frac{dvn(t)}{dt} = \sum F_{vn}(t) + F_{Bvn}(t) + F_{Cvn}(t) \tag{4.7}$$

$$J \cdot \frac{dw(t)}{dt} = \sum T(t) + T_{Bw}(t) + T_{Cw}(t) \tag{4.8}$$

Onde:

- $M\ [Kg]$ - Massa do robot.
- $J\ [Kg \cdot m^2]$ - Momento de inércia do robot.
- $F_v, F_{vn}\ [N]$ - Forças motrizes na direcção de v e vn.
- $T [N \cdot m]$ - Binário de rotação do robot.
- $F_{Bv}, F_{Bvn} [N]$ - Forças de atrito viscoso na direcção de v e vn.
- $T_{Bw} [N \cdot m]$ - Binário de atrito visco na direcção de rotação do robot.
- $F_{Cv}, F_{Cvn} [N]$ - Forças de atrito de Coulomb (estático) na direcção de v e vn.
- $T_{Cw} [N \cdot m]$ - Binário de atrito estático na direcção de rotação do robot.

A resultante das forças e binários motrizes em cada uma das direcções é obtida tendo em conta as forças f_j desenvolvidas por cada um dos motores e a geometria do robot:

$$\sum F_v(t) = (f_3(t) - f_1(t)) \cdot sin(\pi/3) \quad (4.9)$$
$$\sum F_{vn}(t) = -f_2(t) + (f_3(t) + f_1(t)) \cdot cos(\pi/3) \quad (4.10)$$
$$\sum T(t) = (f_1(t) + f_2(t) + f_3(t)) \cdot d \quad (4.11)$$

As forças motrizes são fornecidas pelos três motores, obtidas pelo contacto das rodas com o chão. A força exercida por cada motor pode ser estimada a partir do binário desenvolvido pelo mesmo e pelo raio da roda:

$$f_j(t) = T_j(t)/d \quad (4.12)$$
$$T_j(t) = i_j \cdot l \cdot K_t \quad (4.13)$$

Onde f_j e T_j são a força e binário desenvolvidos pelo motor j, d o raio da roda, i_j a corrente na armadura do motor, l o factor de redução da caixa de velocidades, caso exista, e K_t a constante de binário do motor.

As forças de atrito viscoso são proporcionais à velocidade do robot com constantes de proporcionalidade B_v, B_{vn} e B_w. Temos para cada uma destas:

$$F_{Bv} = -B_v \cdot v(t) \quad (4.14)$$
$$F_{Bvn} = -B_{vn} \cdot vn(t) \quad (4.15)$$
$$T_{Bw} = -B_w \cdot w(t) \quad (4.16)$$

Já as forças de atrito estático são constantes em amplitude, tendo-se:

$$F_{Cv} = -C_v \cdot sign(v(t)) \quad (4.17)$$
$$F_{Cvn} = -C_{vn} \cdot sign(vn(t)) \quad (4.18)$$
$$T_{Cw} = -C_w \cdot sign(w(t)) \quad (4.19)$$

Dada a quantidade de parâmetros que influenciam as constantes C e B (características dos materiais das superfícies em contacto, geometria do robot, geometria das rodas, e afins) não existe nenhum método analítico para as determinar com precisão. Algo semelhante acontece com o momento de inércia J, sendo que a geometria e distribuição de massas do robot tornam o seu cálculo complicado. Estes podem no entanto ser estimados a partir de ensaios com o robot real. Um conjunto de ensaios possível encontra-se descrito em [1] ou [37].

4.5.1 Motor

O robot emprega três motores brushless para motorização. O modelo destes motores é semelhante ao modelo clássico de um motor DC, podendo ser descrito por:

$$u(t) = L \cdot \frac{di(t)}{dt} + R \cdot i(t) + K_v \cdot w_m(t) \tag{4.20}$$

$$T_m(t) = K_t \cdot i(t) \tag{4.21}$$

- $u\,[V]$ - Tensão aos terminais do motor.
- $i\,[A]$ - Corrente na armadura do motor.
- $L\,[H]$ - Indutância do circuito da armadura.
- $R\,[\omega]$ - Resistência do circuito da armadura.
- $K_v\,[V/(rad/s)]$ - Constante de força electromotriz.
- $K_t\,[N \cdot m/A]$ - Constante de binário.
- $w_m\,[rad/s]$ - Velocidade angular do motor.
- $T_m\,[N \cdot m]$ - Binário desenvolvido pelo motor.

Todos os parâmetros do modelo (R, L, K_v e K_t) encontram-se definidos na datasheet do motor. No entanto estes valores variam de motor para motor e modificam-se com o passar do tempo. Podem portanto ser determinados com maior precisão para cada um dos motores por meio de ensaios descritos em [1] ou [37].

4.6 Conclusão

Neste capítulo foram desenvolvidos e apresentados os modelos dinâmico e cinemático do robot omni-direccional utilizado. Foram também fornecidas referências que incluem ensaios para determinação precisa dos parâmetros do modelo dinâmico desenvolvido. O modelo dinâmico completo não é directamente utilizado no controlador desenvolvido, mas é uma referência essencial para a compreensão das simplificações feitas. Já o modelo cinemático é aplicado exactamente como foi descrito para simular a evolução da posição do robot.

Capítulo 5

Introdução ao Controlo Preditivo

Este capítulo pretende dar uma introdução ao controlo preditivo, nas suas variantes linear e não linear. Serão expostos o seu princípio de funcionamento e os resultados teóricos mais importantes, associados a uma formulação matemática do problema. Com o intuito de justificar a sua utilização nos controladores desenvolvidos nesta dissertação serão ainda analisadas criticamente as vantagens e desvantagens do MPC.

5.1 Introdução

O controlo preditivo, do original *Model Predictive Control* ou MPC, surgiu em finais dos anos 70 e tem desde então sofrido um desenvolvimento impressionante, derivado do crescente interesse da indústria e da comunidade científica nesta técnica. Hoje em dia já está suficientemente maduro para ser utilizado em larga escala na indústria, muitas vezes no controlo de processos críticos e exigentes a nível de precisão. Em particular nas duas últimas décadas têm surgido novas aplicações desta estratégia de controlo não só em processos industriais mas também no controlo de robots móveis, de formações dos mesmos, ou mesmo de anestesia clínica, para citar alguns domínios de aplicação. Isto deve-se em grande parte ao crescente poder de processamento das máquinas utilizadas no controlo, o que permite a aplicação do MPC a processos mecânicos, consideravelmente mais rápidos que os processos industriais onde era tradicionalmente aplicado. Em [38, 39] encontra-se uma boa sumarização de técnicas MPC aplicadas à indústria, sendo referidas mais de 2200 aplicações desta estratégia de controlo em processos que vão desde a indústria química à indústria aeroespacial. A aposta da indústria no controlo preditivo deve-se ao facto de este ser capaz de acomodar intrinsecamente as não linearidades do processo e as limitações dos actuadores, ao mesmo tempo que pode incluir as constrições que derivam de preocupações ambientais ou de segurança. De facto o MPC é, como referem *Camacho* e *Bordons* em [16], talvez a forma mais geral de definir um problema de controlo no domínio temporal, integrando controlo óptimo, controlo de processos com *dead time*, controlo multivariável, e utilização de referências futuras. É de

notar que o termo "controlo preditivo"não se refere a uma única estratégia de controlo mas a um vasto leque de algoritmos que fazem uso explícito de um modelo do sistema a controlar e obtêm os valores dos sinais de controlo através da minimização de uma função custo.

5.2 Princípio do Controlo Preditivo

Camacho e *Bordons* apresentam em [16] uma analogia interessante que descreve na perfeição as limitações do controlo puramente reactivo face ao controlo preditivo, relacionando estas estratégias de controlo com a condução de um automóvel. Empregar um controlador reactivo é como conduzir um automóvel olhando apenas para o espelho retrovisor: as acções de controlo só são tomadas depois de o erro ocorrer (i.e. o condutor só começa a virar o automóvel depois de entrar na curva, porque só se apercebe que esta existe quando lá chega). A maneira usual de conduzir um carro é análoga ao funcionamento de um controlador preditivo: o condutor olha para a frente, apercebe-se da estrada que está a seguir e dos carros que lá estão, e toma, de acordo com o modelo que tem do sistema (sabe a maneira como o carro reage), as acções de controlo necessárias para seguir a trajectória correctamente e em segurança (aplicando o travão ou acelerador convenientemente e começando a virar o volante para preparar a curva). Em controlo de processos onde haja informação da referência futura e um modelo preciso do sistema, é clara a vantagem que o controlo preditivo pode fornecer. Os conceitos base afectos a qualquer controlador preditivo são os mesmos e focam-se essencialmente em três pontos, conforme sumarizados em [16]:

- Utilização de um modelo (linear ou não) do sistema para prever a saída do processo em instantes de tempo futuros (horizonte de predição);
- Cálculo de uma sequência de entradas de controlo que minimizam uma dada função objectivo;
- Aplicação da estratégia de horizonte recuante, onde em cada instante de controlo o horizonte de predição é movido em direcção ao futuro e o primeiro sinal de controlo da sequência determinada é aplicado ao sistema.

Por esta razão o controlo preditivo é também tipicamente denominado controlo preditivo de horizonte recuante, do original *Receding Horizon Predictive Control* ou RHCP. De um modo geral, e conforme descrito em [4] por *Findeisen* e *Allgöwer* o princípio básico do MPC é demonstrado pela figura 5.1. Sintetizando conforme [16], o controlo preditivo emprega a seguinte estratégia:

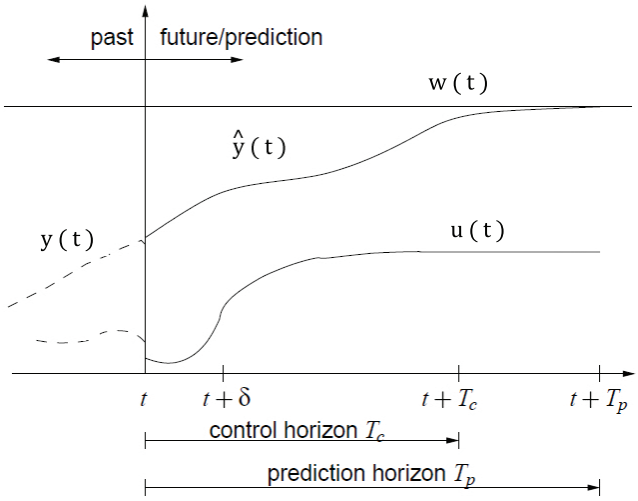


Figura 5.1: Princípio do controlo preditivo (adaptado de [4])

1. Utilizando o modelo do sistema, as saídas $\hat{y}(t)$ do mesmo são previstas para o horizonte de predição T_p definido. Estas saídas dependem do estado actual e passado do sistema e dos sinais de controlo futuros $u(t)$.
2. O conjunto de outputs futuros $u(t)$ é calculado para um horizonte de controlo T_c através da optimização de uma função custo $J(T_c, T_p)$. Esta função custo expressa um determinado critério tal que a saída do processo seja tão próxima da referência $w(t)$ quanto possível. Da sua minimização surgem os valores de $u(t)$ óptimos para o horizonte de controlo T_c.
3. O primeiro valor de $u(t)$ é enviado para o sistema como sinal de controlo e os restantes são rejeitados. O processo completo é repetido no próximo ciclo de controlo, quando novas medidas da saída do sistema estiverem disponíveis.

A optimização da função custo tem diversas particularidades, nomeadamente no que toca ao modo como é realizada. Se o critério utilizado na função custo for quadrático, o modelo for linear, e não houver restrições no sistema, uma solução explícita pode ser encontrada *offline*, passando a implementação por uma simples *lookup table* ou uma função explícita das entradas e saídas passadas (portanto, uma equação às diferenças). Já no caso mais usual de um sistema não-linear, com restrições, ou em que a função custo tome uma forma menos usual, é necessário proceder à optimização *online* em tempo real, utilizando um método numérico. De modo a garantir a convergência da minimização e a obtenção de sinais controlo suaves (em determinadas aplicações, como por exemplo a actuação de válvulas mecânicas, um controlo "nervoso"não é desejável) é habitual incluir também na função custo um termo que penaliza o esforço de controlo.

Esta estratégia de controlo é tipicamente implementada recorrendo à estrutura básica apresentada no modelo da figura 5.2.

Ilustrados estão os dois principais elementos de um controlador preditivo: o optimizador e o modelo do sistema. O modelo utiliza a informação passada do sistema para prever a resposta futura

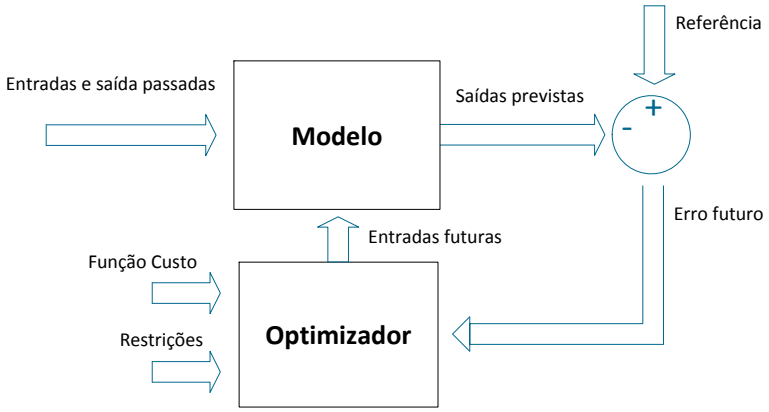


Figura 5.2: Modelo genérico de um controlador preditivo (adaptado de [4])

deste a determinadas entradas de controlo. Estas são por sua vez calculadas pelo optimizador, que minimiza os erros futuros optimizando a função custo para obter os sinais de controlo ideias. As restrições do sistema são tidas em conta nesta optimização.

5.3 Controlo Preditivo Linear e Não Linear

O controlo preditivo linear - *Linear MPC* - caracteriza-se por uma família de estratégias de controlo preditivo em que o modelo dinâmico utilizado para prever a evolução do sistema é linear, ainda que a dinâmica do sistema em malha fechada seja não linear devido à existência de restrições. Esta simplificação é feita garantindo que o sistema opera numa zona controlada onde pode efectivamente ser linearizado. Há no entanto muitos sistemas que são inerentemente não lineares, caso em que um modelo não linear tem que ser obrigatoriamente utilizado. Este facto, aliado às crescentes preocupações de ordem económica e ambiental que levam a que os sistemas tenham muitas vezes que ser operados perto dos limites da região admissível, levou ao desenvolvimento e uso de controlo preditivo não linear - *Nonlinear Model Predictive Control*, ou NMPC.

A natureza não linear dos modelos dinâmicos utilizado em NMPC põe fora de questão a obtenção de uma solução explícita para a optimização da função custo. Esta tem que ser realizada recorrendo a métodos numéricos iterativos, tipicamente baseados em técnicas de minimização Newtonianas, como gradiente descente (para uma boa referência sobre técnicas de optimização consultar [25]). O controlo preditivo não linear levanta alguns problemas de convergência e estabilidade que são consideravelmente mais pertinentes do que na sua versão linear. *Fontes* apresenta em [40] uma *framework* generalizada para projectar controladores NMPC estáveis para sistemas não lineares com restrições nas entradas.

5.4 Formulação matemática do NMPC

De seguida apresenta-se uma formulação matemática contínua do NMPC (a variante de controlo preditivo utilizada nos controladores desenvolvidos neste projecto), conforme exposta em [4] por *Findeisen et Al.*

Considere-se a classe de sistemas não-lineares definidos pelo seguinte conjunto de equações diferenciais:

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(0) = x_0 \tag{5.1}$$

Sujeita a restrições de estado e entrada na forma:

$$u(t) \in U \forall t \geq 0 \quad e \quad x(t) \in \chi \vee t \geq 0, \tag{5.2}$$

Onde $x(t) \in \chi^n \subseteq \mathbb{R}^n$ é o vector de estado e $u(t) \in U \subseteq \mathbb{R}^m$ é o vector de entradas do sistema. O conjunto de estados atingíveis do sistema é dado por χ e o conjunto de entradas válidas por U. Assume-se que U é compacto, χ é conexo e $(0,0) \in \chi \times U$. As restrições a que os conjuntos de entradas e estados estão sujeitos podem ser definidas na sua forma mais simples por:

$$\chi = \{ x \in \mathbb{R}^n \mid x_{min} \geq x \geq x_{max} \} \tag{5.3}$$

$$U = \{ u \in \mathbb{R}^m \mid u_{min} \geq u \geq u_{max} \} \tag{5.4}$$

Onde u_{max}, u_{min}, v_{max}, v_{min} são vectores constantes.

De modo a distinguir claramente o sistema real do simulado utilizado para prever a evolução do estado, as variáveis internas do controlador são diferenciadas por uma barra ($\bar{x}$, por exemplo). O problema de controlo óptimo em horizonte finito descrito anteriormente pode então ser formulado matematicamente da seguinte maneira:

Determinar

$$min_{\,\bar{u}(\cdot)}\, J(x(t), \bar{u}(\cdot); T_c, T_p) \tag{5.5}$$

com

$$J(x(t), \bar{u}(\cdot); T_c, T_p) = \int_t^{t+T_p} F(\bar{x}(\tau), \bar{u}(\tau)) d\tau \tag{5.6}$$

sujeito a

$$\bar{x}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(t) = x(t) \tag{5.7}$$

$$\bar{x}(\tau) \in \chi, \quad \vee\tau \in [t, t+T_p] \tag{5.8}$$

$$\bar{u}(\tau) = \bar{u}(\tau + T_c), \quad \forall\tau \in [t+T_c, t+T_p] \tag{5.9}$$

$$\bar{u}(\tau) \in U, \vee\tau \in [t, t+T_c] \tag{5.10}$$

onde T_p e T_c são os horizontes de predição e controlo, respectivamente, sendo que $T_c \leq T_p$. Conforme descrito atrás, a barra indica as variáveis internas do controlador. $\bar{x}(\cdot)$ é a resposta de 5.7 à entrada $\bar{u}(\cdot) : [t, t+T_p]$. Os valores previstos, mesmo se for considerado um sistema sem perturbações, nunca são os valores reais em malha fechada, dado que a entrada óptima do sistema é recalculada em cada ciclo de controlo ao longo de um horizonte de controlo recuante T_c.

5.4.1 Função Custo

A função custo F utilizada em 5.6 define os critérios de penalização que são minimizados de forma a calcular a entrada óptima do sistema. Estes podem dizer respeito não só ao sistema directamente mas podem também surgir de restrições económicas, ambientais, ou de segurança. A função custo toma habitualmente a forma quadrática *standard* definida na equação seguinte:

$$F(x,u) = (x - x_s)^T Q (x - x_s) + (u - u_s)^T R (u - u_s) \tag{5.11}$$

onde x_s e u_s são os *setpoints* para o estado e entradas do sistema, respectivamente. Q e R são matrizes quadradas positivas e simétricas denominadas matrizes de pesos. Estas contêm o peso a dar a cada penalização contida na função custo. O primeiro elemento da função custo definida em 6.2 visa penalizar o desvio do estado do sistema da referência pretendida. O segundo é normalmente utilizado para penalizar a variação do esforço de controlo, caso em que u_s corresponde aos valores das entradas de controlo no instante de tempo anterior.

5.4.2 Solução óptima

De modo a garantir que a referência desejada (x_s, y_s) seja uma solução possível do problema de minimização é necessário garantir que $u_s \in U$. O modelo do sistema utilizado para prever a evolução do estado é, conforme indicado pela equação 5.7, inicializado em cada instante de controlo para o estado do sistema real.

A solução óptima para o problema de optimização é representada por $\bar{u}^*(\tau; x(t), T_p, T_c) : [t, t+T_p] \rightarrow U$. O problema de optimização em malha aberta é resolvido repetidamente a cada instante de tempo em que novas medidas do estado do sistema sejam disponibilizadas. O sinal controlo

em malha fechada u é definido pela solução óptima do problema da equação 5.5 desde o instante actual até aos instantes de tempo de disponibilização de novas medidas δ:

$$u^*(\tau) = \bar{u}^*(\tau; x(t), T_p, T_c), \tau \in [t, \delta] \tag{5.12}$$

O valor óptimo do problema de minimização é dado por V, sendo definido como:

$$V(x; T_p, T_c) = J(x, \bar{u}^*(\tau; x(t)); T_p, T_c). \tag{5.13}$$

5.4.3 Horizontes de Controlo e Predição

Seria desejável a utilização de horizontes de controlo e predição infinitos ($T_c = T_p = \infty$) de modo a garantir a minimização dos critérios de penalização definidos pela função custo e a obtenção dos sinais de controlo óptimos. No entanto, uma vez que a optimização tem que ser realizada em tempo real recorrendo a métodos numéricos é imperativo que se usem horizontes temporais finitos. De facto, de um ponto de vista de tempo computacional, quanto mais pequenos forem os horizontes de predição utilizados menor tempo de execução será necessário para o algoritmo de controlo e mais adequado será este para utilização em tempo real.

Daqui derivam alguns resultados importantes. Com um horizonte de predição finito as entradas e a evolução do estado do sistema em malha fechada vão sempre ser diferentes das previstas pelo controlador em malha aberta, mesmo que o modelo seja perfeito e não existam perturbações. Isto significa que a minimização do custo ao longo de um horizonte temporal infinito nunca é atingida ([4]). Portanto, se a evolução real e prevista do estado do sistema diferem, não existem garantias que o sistema seja estável em malha fechada. Nestes casos torna-se necessário que a função custo penalize não só as grandezas que dizem directamente respeito aos objectivos de controlo a atingir mas que também contenha critérios que ajudem a garantir a estabilidade (nomeadamente, penalizando o esforço de controlo ou a variação do mesmo). Resultados mais vastos sobre a estabilidade e robustez de controladores NMPC podem ser encontrados em [40] ou [41].

5.5 Vantagens e Desvantagens

As vantagens do MPC quando comparado com outras estratégias de controlo menos sofisticadas são claras. De entre todas, consideram-se as mais pertinentes:

- Pode ser utilizado para controlar uma grande variedade de processos, desde aqueles com dinâmica relativamente simples a processos com dinâmica complexa, atrasos longos, ou muito instáveis.
- O caso de processos com controlo multivariável é facilmente tratado, sendo intrínseco ao controlador.
- Tem, intrinsecamente, compensação de *dead time*.

- O tratamento de restrições é simples, dado que estas são incluídas naturalmente na fase de projecto do controlador.
- É extremamente útil nos casos em que as referências futuras estejam disponíveis (como o caso de seguimento de trajectórias por robots móveis).
- É uma técnica relativamente recente que suscita muito interesse tanto da comunidade académica como da indústria, o que resulta num desenvolvimento rápido e consistente.

Naturalmente, o MPC também apresenta algumas desvantagens. Estas prendem-se essencialmente com a maior complexidade de implementação de estratégias de controlo preditivo face a técnicas de controlo clássico, aliado a um maior peso computacional do controlador. Esta última é especialmente importante quando se trata de aplicações industriais, onde os computadores utilizados são não raras vezes pouco impressionantes no que toca a poder de processamento. Há ainda a considerar necessidade de ter um modelo dinâmico preciso do sistema a controlar, o que pode ser difícil de obter no caso de sistemas altamente complexos.

5.6 Conclusão

Neste capítulo foi apresentada uma breve introdução ao controlo preditivo. Foram referidos os princípios subjacentes a este tipo de controlo, assim como a estratégia genérica utilizada para implementar um controlador preditivo. Foi feita a diferenciação entre as suas variantes linear e não linear, sendo apresentada uma formulação matemática para o NMPC. Finalmente, foram discutidas as vantagens e desvantagens desta estratégia do MPC, numa tentativa de justificar a sua utilização neste projecto.

O próximo capítulo detalha o projecto e implementação de um controlador do tipo NMPC para seguimento de trajectórias por um robot móvel, tendo em conta as bases teóricas aqui apresentadas.

Capítulo 6

Controlador de Trajectória

Este capítulo detalha a estrutura, funcionamento, e implementação do controlador preditivo para controlo do seguimento de trajectórias desenvolvido nesta dissertação. São apresentados os métodos utilizados para calibração e optimização do controlador, assim como resultados ilustrativos do seu desempenho.

6.1 Introdução

Antes que se possa sequer equacionar a hipótese de projectar e implementar um bom controlador de formações para uma equipa de robots móveis é primeiro necessário que cada robot individual possua um controlador para seguimento de trajectórias com um desempenho apreciável. Isto torna-se particularmente importante quando se considera o seguimento de trajectórias a velocidades elevadas, com mudanças bruscas de sentido ou orientação, e em ambientes extremamente dinâmicos (como é o exemplo de um jogo de futebol robótico). O Capítulo 2.2 resumiu algumas das estratégias mais recentemente desenvolvidas para seguimento de trajectórias. Do estudo das soluções apresentadas conclui-se que um controlador reactivo dificilmente atingiria o desempenho desejado, tendo em conta as velocidades desejadas e as características das trajectórias em jogo. A escolha recai portanto sobre os controladores do tipo preditivo.

Conforme foi concluído em 2.4, as características específicas dos controladores preditivos tornam-nos uma opção a ter em conta para o projecto de controladores de trajectória onde o controlo eficiente a altas velocidades é requerido. O controlador descrito em [1], [23] e [24] é uma solução particularmente interessante. Foi desenvolvido no âmbito de um projecto semelhante, também para os robots de uma equipa de futebol robótico (omni-direccionais de 4 rodas, ao invés dos robots de 3 rodas utilizados nesta dissertação), e emprega métodos de optimização numérica que o tornam passível de ser utilizado para controlo em tempo real. Os excelentes resultados obtidos pelo autor do controlador são também eles um bom indicador de que esta é uma boa opção.

Optou-se então pela implementação de um controlador baseado no trabalho referido acima, adaptando a sua estrutura às necessidades actuais. Existe uma grande diferença entre o controlador implementado nesta dissertação e aquele em que este se baseou. Em [1], *Scolari* começou por construir um modelo exaustivo do robot, determinando com precisão todos os parâmetros relevantes através de testes com o robot real. Este modelo extremamente complexo foi utilizado no controlador para prever a evolução do estado do sistema. Uma modelação tão exaustiva do robot sai fora do âmbito desta dissertação, sendo a sua complexidade quase suficiente para justificar a sua própria tese de mestrado. Neste caso, tomou-se a aproximação inversa: começou-se com um modelo muito simples do robot, unicamente cinemático, e foram-se adicionando elementos que tornam o modelo mais realista (resposta dinâmica e saturação dos motores, limites de velocidade, e afins) até que o controlador apresentasse um desempenho satisfatório. Dada a quantidade de simplificações envolvidas, o modelo final utilizado no controlador não corresponde com precisão ao robot real, mas apresenta uma resposta suficientemente semelhante para que o controlador funcione correctamente, como mais tarde será verificado.

6.2 Definição do Sistema

É necessária uma definição matemática do sistema a controlar, sistematizando entradas, saídas, e variáveis correspondentes ao estado do sistema. Este trata-se de um robot móvel omnidireccional de três rodas, utilizado em futebol robótico. O robot já possui controladores de velocidade para cada um dos motores e o software já contém métodos que permitem fornecer referências de velocidades linear e angular, pelo que o controlador irá operar no nível acima deste (fornecendo essas referências de velocidade).

O estado do robot corresponde à sua posição no mundo, no sistema de coordenadas globais (XoY). É representado pelo vector $X(k) = [\, x(k)\ y(k)\ \theta(k)\,]^T$, onde $x(k)$ e $y(k)$ correspondem à posição e $\theta(k)$ à orientação do robot, no sistema de coordenadas global. As entradas do sistema são as velocidades de referência no sistema de coordenadas do robot, definidas pelo vector $U(k) = [\, v(k)\ vn(k)\ w(k)\,]^T$. $v(k)$ é a velocidade linear pretendida na direcção da orientação do robot, $vn(k)$ a velocidade linear pretendida na direcção perpendicular à anterior, e $w(k)$ a velocidade angular. Os parâmetros aqui descritos podem ser observados na figura 6.1.

A trajectória de referência é representada por $R(i) = [\, x(i)\ y(i)\ \theta(i)\,]^T$. Analogamente ao vector de estado do robot, $x(i)$, $y(i)$, e $\theta(i)$ representam as posições e orientação desejadas para o robot. Convém relembrar que esta trajectória não está parametrizada em termos de tempo, correspondendo apenas à sequência de pontos a seguir. A dimensão tempo é considerada mais tarde, tendo em conta o módulo da velocidade desejado.

Dos vectores de entradas e estado do sistema retiramos a equação de evolução do estado do robot:

$$X(k+1) = f(X(k), U(k)) \tag{6.1}$$

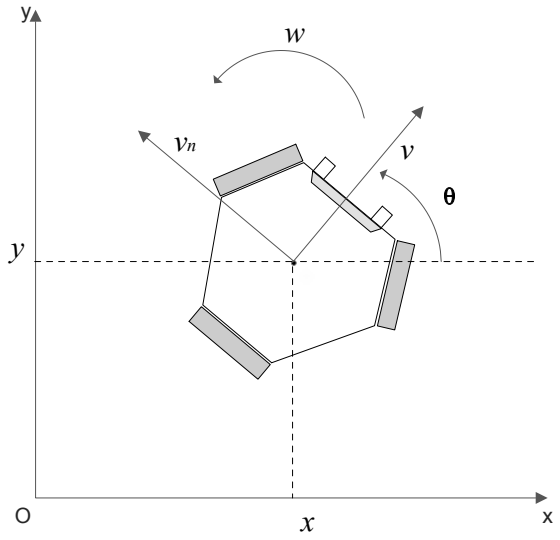


Figura 6.1: Esquema do estado do robot e sistemas de coordenadas

No âmbito da simulação são consideradas mais duas estruturas:

- $X_{sim}(k) = [\, x_{sim}(k)\ y_{sim}(k)\ \theta_{sim}(k)\,]^T$ - Estado da simulação interna do robot, vai evoluindo conforme vai sendo previsto pelo simulador. No início de cada iteração do ciclo de controlo é definido para o estado actual do robot real.
- $R_{ref}(k) = [\, x_{ref}(k)\ y_{ref}(k)\ \theta_{ref}(k)\,]^T$ - Trajectória de referência para o controlador, utilizada no simulador para calcular o valor da função custo. É calculada a partir de $R(k)$ para o horizonte de predição N_p, tendo em conta a posição do robot no início do ciclo de controlo $X_{sim}(0)$ e o módulo da velocidade pretendido $|V_{ref}|$. Os pontos da trajectória correspondem à posição do robot caso este seguisse idealmente a trajectória pretendida à velocidade de referência.

6.3 Estrutura do Controlador

O conceitos básicos do controlador são bastante simples e intuitivos e derivam directamente da formulação clássica do MPC: tendo-se um modelo do robot, prevê-se a evolução do seu estado (posição e orientação no mundo) para um determinado intervalo de tempo como resposta a diversas entradas de controlo. O estado do modelo interno do robot é inicializado a cada ciclo do algoritmo de controlo para o estado do robot real. Uma função custo é utilizada para, por comparação da evolução do estado prevista com a trajectória desejada, calcular um custo associado a cada conjunto de entradas de controlo. A função custo penaliza essencialmente as diferenças entre a posição e orientação do robot previstas e as desejadas para cada instante de tempo. Da optimização (entenda-se, minimização) desta função custo encontram-se as entradas de controlo óptimas a aplicar ao robot.

O controlador é constituído por três blocos fundamentais: *Optimizador*, *Simulador*, e *Gerador de trajectória de referência*, sendo a sua estrutura esquematizada na 6.2. Os três blocos têm funções muito específicas:

- **Gerador de trajectória de referência** - A partir da trajectória pretendida R, do módulo da velocidade desejada $|V|$, e do estado do modelo de simulação X_{sim}, gera a trajectória de referência R_{ref} que é utilizada para cálculo do valor da função custo.
- **Simulador** - Contém o modelo do robot e a definição da função custo. O estado do robot simulado X_{sim} é inicializado para o estado do robot real X de cada vez que o ciclo de controlo de inicia. O simulador prevê, para um determinado conjunto de entradas de controlo U, a evolução do estado do robot para o horizonte de predição desejado. Por comparação das posições e orientações previstas com a trajectória de referência R_{ref} é calculado um custo J associado a estas entradas.
- **Optimizador** - Utiliza um método de gradiente descendente para minimizar o valor da função custo, tendo em conta os valores de custo J fornecidos pelo simulador para cada conjunto de entradas U. As entradas óptimas calculadas são aplicadas ao robot.

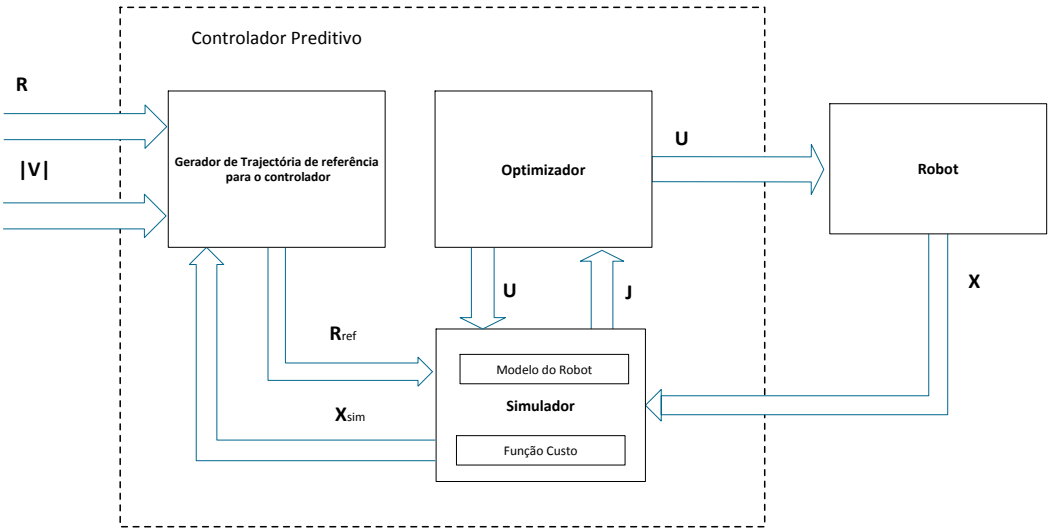


Figura 6.2: Estrutura do controlador de trajectória

Dois conceitos importantes são definidos como parâmetros do controlador: o *horizonte de predição*, T_p, e o *horizonte de controlo*, T_c. O horizonte de predição corresponde ao intervalo de tempo para o qual se prevê a evolução do estado do robot, a partir do estado actual. O horizonte

de controlo é o intervalo de tempo, a partir do estado actual, para o qual se prevê as entradas de controlo optimizadas para o robot. Considerando que se trabalha em tempo discreto, tem-se:

- N_p - Horizonte de predição, em número de ciclos do controlador (sendo que um ciclo é executado a cada 40ms). Número de ciclos para qual a evolução do estado do robot é prevista.
- N_c - Horizonte de controlo, também em número de ciclos do controlador. Número de instantes de controlo diferentes que são aplicados durante a simulação.

6.4 Algoritmo de Controlo

O algoritmo de controlo é executado com um período de $40ms$, uma vez que está sincronizado com a chegada de imagens da câmara utilizada para localizar o robot no campo (que fornece imagens a 25 *FPS*). Deste período de $40ms$, uma considerável parte do tempo é utilizado para tarefas de processamento de imagem, pelo que o tempo total de execução do algoritmo deve ser bastante inferior. O diagrama de fluxo da figura 6.3 descreve o funcionamento do algoritmo implementado.

6.5 Definição da Função Custo

A função custo é um elemento essencial do controlador, na medida em que fornece uma medida da qualidade de cada conjunto de entradas de controlo. É da minimização da função custo, por métodos numéricos, que são encontradas as entradas óptimas a aplicar ao sistema.

Porque se trata de um problema de seguimento de trajectórias, a função custo deve obviamente penalizar as diferenças entre as posições e a orientações do robot e as posições e orientações de referência. Adicionalmente, a função custo contém um termo que penaliza a variação das entradas de controlo, de forma a minimizar o esforço de controlo. A cada uma destas penalizações está associado um peso, que define a proporção da penalização no valor global da função custo. Assim, a função de custo encontra-se definida na equação 6.2 como:

$$\begin{aligned} J(N_1,N_2,N_c) = & \sum_{i=N_1}^{N_2} (\lambda_1([x_{sim}(i)-x_{ref}(i)]^2+[y_{sim}(i)-y_{ref}(i)]^2))+ \\ & \sum_{i=N_1}^{N_2} (\lambda_2[\theta_{sim}(i)-\theta_{ref}(i)]^2)+ \\ & \sum_{i=1}^{N_c} \lambda_3(\Delta U(i)) \end{aligned} \tag{6.2}$$

Onde:

- N_1 e N_2 são os limites do horizonte de predição que se pretende incluir no cálculo da função custo tal que $N_1 > 0$ e $N_2 \leq N_p$.

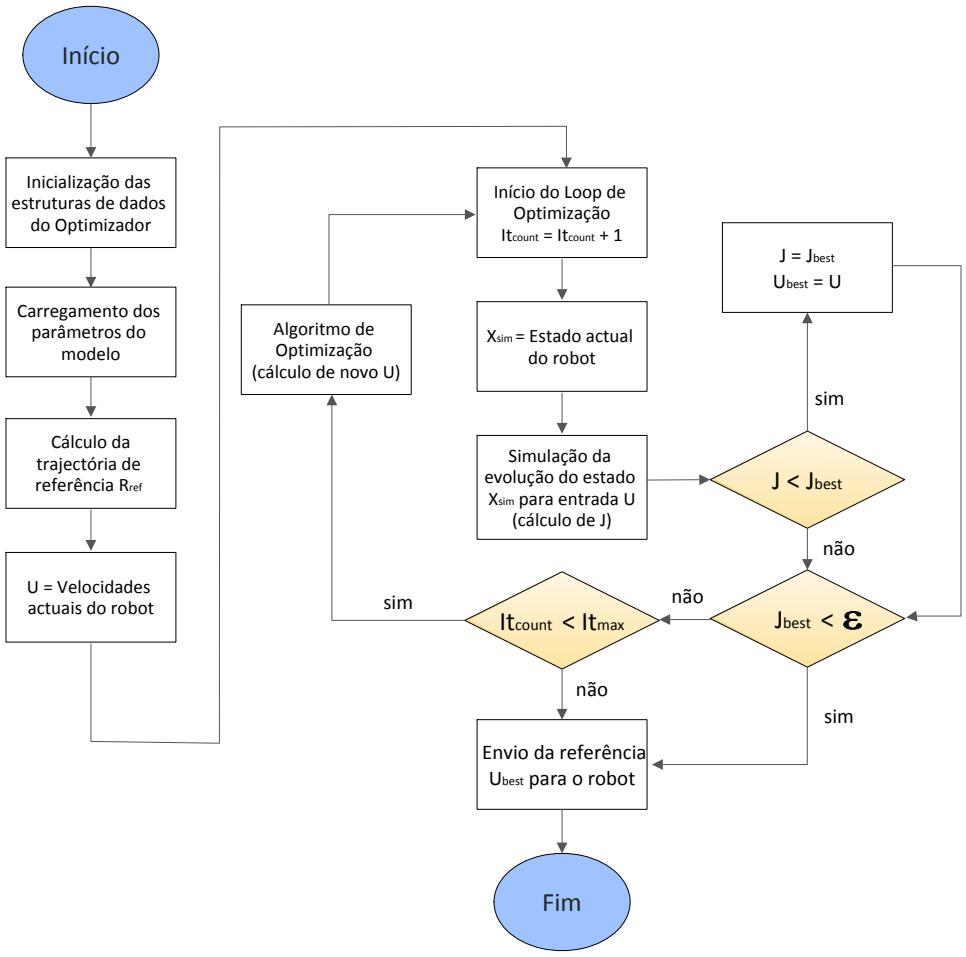


Figura 6.3: Diagrama de fluxo do algoritmo de controlo de trajectória

- N_c é o horizonte de controlo.
- λ_1, λ_2, e λ_3 são os pesos associados a cada um dos componentes da função custo.
- $x_{sim}(k)$, $y_{sim}(k)$, e $\theta_{sim}(k)$ são os elementos do vector $X_{sim}(k)$, correspondendo à posição prevista do robot no instante i.
- $x_{ref}(k)$, $y_{ref}(k)$, e $\theta_{ref}(k)$ são os elementos do vector $R_{ref}(k)$, correspondendo à posição de referência do robot para o instante i.
- $\Delta U(k) = [v(k)_{ref} - v_{ref}(k-1)]^2 + [vn_{ref}(k) - vn_{ref}(k-1)]^2 + [w_{ref}(k) - w_{ref}(k-1)]^2$, correspondendo à variação dos valores de controlo, sendo $U(i) = [\ v_{ref}(k)\ vn_{ref}(k)\ w_{ref}(k)\ ]^T$ o vector das velocidades de referência.

6.6 Cálculo da Trajectória de Referência para o Controlador

A trajectória de referência para o robot, $R(k) = [\, x(k)\ y(k)\ \theta(k)\,]^T$, não é mais do que um conjunto ordenado de pontos e orientações definidos em coordenadas do sistema global, sem qualquer parametrização temporal, e que não necessitam sequer de estar igualmente espaçados entre si. Tome-se como exemplo a figura 6.4.

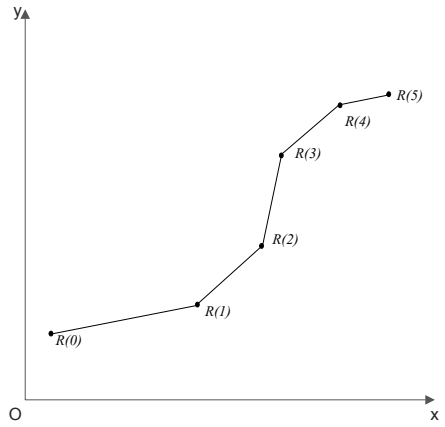


Figura 6.4: Trajectória de referência, com $R(k) = [\, x(k)\ y(k)\ \theta(k)\,]^T$

Para calcular o valor da função custo é necessário calcular a partir de $R(k)$ uma nova trajectória de referência $R_{ref}(k) = [\, x_{ref}(k)\ y_{ref}(k)\ \theta_{ref}(k)\,]^T$, cujos pontos representem a posição e orientação desejada para o robot nos instantes $k \in [0, N_p]$. Estas serão as posições e orientações de referência para o robot nos próximos N_p instantes, separados por $40ms$ (correspondente ao período do algoritmo de controlo principal do robot). $R_{ref}(k)$ é então calculada a partir de $R(k)$, $X(0)$ (estado do robot no início do ciclo de controlo), e $|V|$ (o módulo da velocidade pretendida). A figura 6.5 ilustra o procedimento executado.

6.6.1 Cálculo de x_{ref} e y_{ref}

As coordenadas do primeiro ponto da trajectória de referência para o controlador, $R_{ref}(0)$, são obtidas pela intersecção do segmento de recta da trajectória $R(k)$ mais próximo da posição actual do robot com a recta perpendicular a esse segmento e que passa nessa mesma posição (ver figura 6.5). Por outras palavras, é o ponto dos segmentos que compõem a trajectória global mais próximo do robot.

As coordenadas dos restantes pontos de $R_{ref}(k)$ são obtidas sequencialmente calculando pontos pertencentes à trajectória $R(k)$ tal que:

$$dist(R_{ref}(i), R_{ref}(i-1)) = |V| * 0.04 \tag{6.3}$$

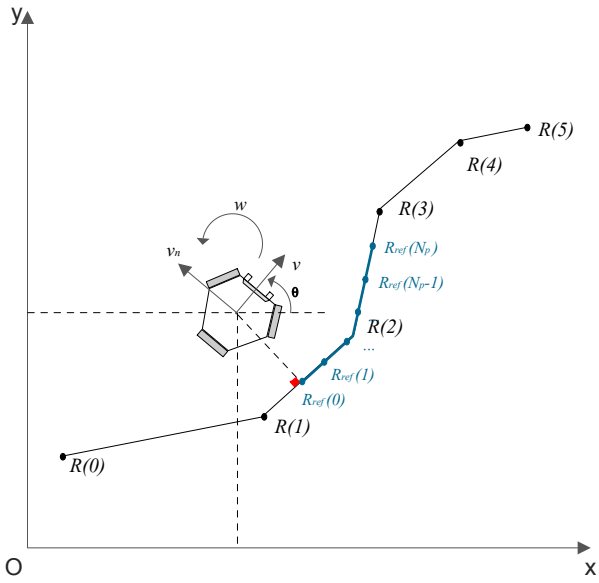


Figura 6.5: Obtenção da trajectória de referência para o controlador

Na equação 6.3, $dist(a,b)$ é a distância entre os pontos a e b, $|V|$ é o módulo da velocidade pretendida em m/s, e 0.04 é o período do ciclo de controlo, em s. No caso de a trajectória global mudar de segmento entre dois pontos de $R_{ref}(k)$, ou seja, de existir um ponto $R(j)$ entre $R_{ref}(i)$ e $R_{ref}(i+1)$, o novo ponto $R_{ref}(i+1)$ é calculado sobre o segmento de recta definido por $R(i)$ e $R(i+1)$ tal que:

$$dist(R_{ref}(i),R(j)) + dist(R_{ref}(i+1),R(j)) = |V| * 0.04 \tag{6.4}$$

Este caso encontra-se exemplificado na figura 6.5, onde a trajectória de referência para o controlador foi calculada ao longo de dois segmentos de $R(k)$.

6.6.2 Cálculo de θ_{ref}

A referência de orientação θ_{ref} para $R_{ref}(0)$ corresponde à orientação inicial do robot. Para cada um dos pontos restantes de $R_{ref}(k)$, a referência de orientação é calculada por interpolação linear do valor das referências de orientação dos pontos da trajectória completa $R(k)$:

$$\theta_{ref}(i+1) = \theta_{ref}(i) + \theta_{step} \tag{6.5}$$

Para cada segmento de $R(k)$, θ_{ref} deve ser interpolado entre as referências de orientação dos pontos que o definem. O valor de θ_{step} é calculado para cada um dos segmentos da trajectória completa, dependendo da velocidade de referência, do tempo do ciclo de controlo, e das referências

de orientação dos extremos do segmento. Sistematizando, para o segmento definido por $R(m-1)$ e $R(m)$ temos:

$$\theta_{step} = \frac{\theta(m) - \theta(m-1)}{\frac{dist(R(m),R(m-1))}{|v|*0.04}} \tag{6.6}$$

Salvo raros acasos, a trajectória de referência calculada irá começar a meio de um segmento da trajectória completa. Nesse primeiro segmento, θ_{ref} deve ser interpolado entre a orientação actual do robot e a referência de orientação do ponto extremo desse segmento (no caso da figura 6.5, entre θ actual do robot e a orientação de referência do ponto $R(2)$). Assumindo que os primeiros pontos da trajectória são marcados no segmento de $R(k)$ definido por $R(j-1)$ e $R(j)$ tem-se:

$$\theta_{step} = \frac{\theta_{ref}(0) - \theta(j)}{\frac{dist(R_{ref}(0),R(j))}{|v|*0.04}} \tag{6.7}$$

6.7 Modelo Simplificado do Sistema

A grande diferença do controlador projectado nesta tese em relação ao projectado por *Scolari* (ver [1]) reside no modelo simplificado utilizado. Ao invés de obter e parametrizar um modelo minucioso do robot começou-se por utilizar primeiro um modelo unicamente cinemático (descrito em 4.4) e ir adicionando elementos dinâmicos até o controlador apresentar o desempenho desejado. A utilização de um modelo simplificado, quando correctamente parametrizado, traz vantagens no sentido de reduzir a carga computacional de cada ciclo do algoritmo de controlo. Isto traduz-se na possibilidade de utilizar horizontes de predição e de controlo maiores para o controlador preditivo, mesmo em computadores com especificações algo modestas, mantendo o tempo do ciclo de controlo dentro dos limites requeridos.

O modelo utilizado na versão final do controlador consiste num modelo cinemático ao qual foram adicionados dois elementos: limite de velocidade de rotação dos motores, com detecção de saturação e escalonamento proporcional das velocidades dos restantes motores, e aproximação da resposta ao degrau de cada motor por um sistema linear invariante de primeira ordem. Ambos serão descritos de seguida.

6.7.1 Limitação da velocidade dos motores

De modo a ter em conta a limitação de velocidade dos motores foi implementado na simulação o algoritmo IRV (*Ideal Reference Velocities*) apresentado por *Scolari et Al.* em [12]. A motivação por detrás do desenvolvimento deste algoritmo baseia-se no facto de que se uma determinada referência de velocidade fornecida ao robot saturar um ou mais motores e as velocidades dos restantes não forem modificadas, o robot vai seguir na direcção errada. Isto resulta, claro, numa quebra no desempenho do controlador devido ao afastamento do robot da trajectória de referência. O que o IRV faz é detectar se alguma das referências de velocidade fornecidas aos motores é superior à velocidade máxima dos mesmos, e calcular a partir daí um factor de escalonamento

para redefinir as velocidades de todos os motores, garantindo a correcta direcção de movimento do robot. Da implementação deste algoritmo na simulação garante-se que ao robot real nunca serão fornecidas referências de velocidade que saturem os motores.

Sejam V_1, V_2, V_3 as referências de velocidade para cada um dos motores e V_{limit} o limite de velocidade de um motor. O processo utilizado encontra-se no algoritmo 1.

Algorithm 1 Ideal Reference Velocities

$V_{max} = max(V_1, V_2, V_3)$
$V_{min} = min(V_1, V_2, V_3)$
if $V_{max} > V_{limit}$ **then**
  $Scale_{max} = V_{max}/V_{limit}$
else
  $Scale_{max} = 1$
end if
if $V_{min} < -V_{limit}$ **then**
  $Scale_{min} = V_{min}/-V_{limit}$
else
  $Scale_{min} = 1$
end if
$Scale = max(Scale_{max}, Scale_{min})$
$V_1 = V_1/Scale$
$V_2 = V_2/Scale$
$V_3 = V_3/Scale$

6.7.2 Resposta Dinâmica dos Motores

Conforme descrito, o driver de cada um dos motores contém um controlador PID para seguimento de referências de velocidade. Este conjunto de motor + controlador PID foi modelizado por um sistema LTI (*Linear Time Invariant*) de primeira ordem, aproximação que embora um pouco distante do comportamento do real do sistema é suficientemente próxima para que, uma vez correctamente parametrizada, forneça resultados muito satisfatórios. Um sistema deste tipo apresenta uma resposta invariante ao degrau, o que permite simplificar a simulação.

Seja $u(t)$ a referência de velocidade para o motor e $y(t)$ a velocidade real. Um sistema LTI de primeira ordem para este caso pode ser definido pela equação 6.8:

$$\frac{dy(t)}{dt} + \frac{1}{\tau} \cdot y(t) = K \cdot u(t) \tag{6.8}$$

Aplicando a transformada de Laplace obtém-se 6.9:

$$s \cdot Y(s) + \frac{1}{\tau} \cdot Y(s) = K \cdot U(s) \tag{6.9}$$

Podemos então definir uma função de transferência para o sistema (equação 6.10):

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{s+r} \tag{6.10}$$

Onde $r = 1/\tau$, sendo τ a constante de tempo do sistema e K o seu ganho DC. Um sistema deste tipo apresenta uma resposta exponencial ao degrau, da forma descrita na equação 6.11 (assumindo $u(t)$ constante para $t \geq 0$ e $K = 1$).

$$y(t) = e^{-t/\tau} \cdot (y(0) - u(t)) + u(t) \tag{6.11}$$

Os resultados apresentados em [42] foram utilizados para implementar uma função de transferência digital, descrita na equação 6.12.

$$\begin{aligned} y(k) =& a_1 \cdot y(k-1) + b_1 \cdot u(k-1) \\ a_1 =& e^{-rT} \\ b_1 =& (1/r) \cdot [-e^{-rT}] \end{aligned} \tag{6.12}$$

Aqui, $y(k)$ corresponde à velocidade do motor no instante k, $u(k)$ à referência de velocidade no mesmo instante, T o tempo de amostragem do sistema discreto, e $r = 1/\tau$, com τ sendo a constante de tempo do sistema. A velocidade do motor no instante k é portanto calculada a partir da sua velocidade e referência de velocidade no instante anterior e da constante de tempo definida para o sistema. O gráfico da figura 6.6 foi obtida por aplicação desta equação, e caracteriza o tipo de resposta exponencial à qual a resposta ao degrau dos motores foi aproximada.

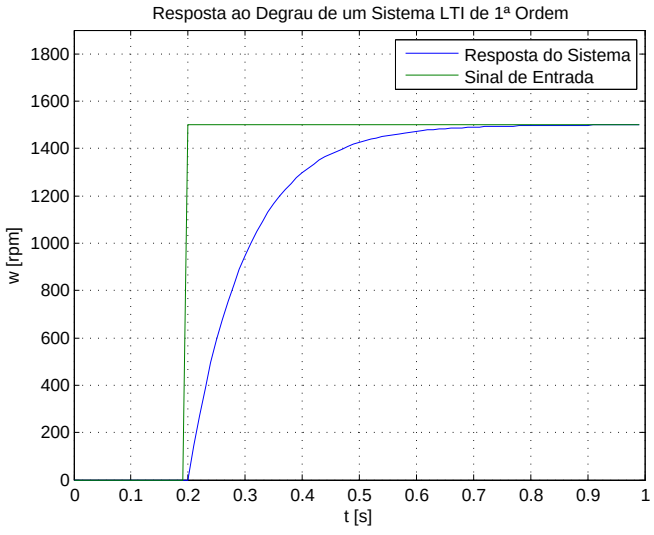


Figura 6.6: Resposta ao degrau de um sistema LTI de 1ª ordem, com $\tau = 0.1$ e $K = 1$

6.8 Simulação da Evolução do Sistema

O processo de simulação da evolução do sistema é utilizado para prever, para o horizonte de predição desejado N_p, a evolução do estado do sistema (entenda-se, da posição e orientação do

robot) como resposta a diferentes entradas de controlo. É durante este processo de simulação que é calculado o custo associado às entradas de controlo utilizadas. Considera-se um instante de simulação a determinação da posição do robot no instante de controlo seguinte (ou seja, $T = 0.04s$ segundos depois) em relação ao instante inicial. A simulação completa para um horizonte de predição N_p corresponde então a N_p simulações, ou a uma predição da evolução do sistema para os próximos $N_p * 0.04s$.

Esta simulação é feita por aplicação do modelo descrito na secção 6.7, e processa-se de acordo com o esquematizado na figura 6.7. Do estado actual do robot $X_{sim}(k)$ e das velocidade de referência pretendidas, obtém-se o estado no instante seguinte $X_{sim}(k+1)$.

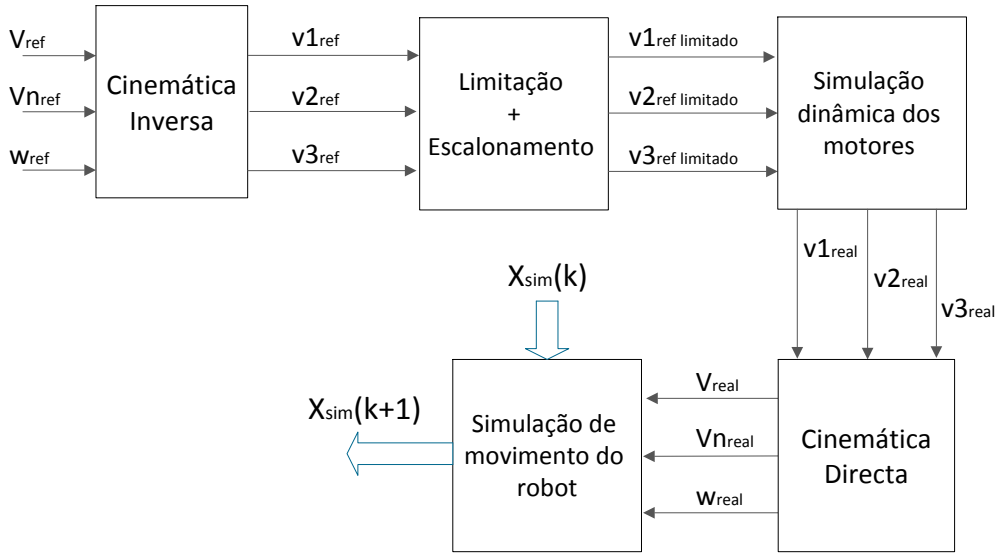


Figura 6.7: Processo de simulação da evolução do sistema

A figura 6.7 divide o processo de simulação nas diversas etapas que o compõem. Cada uma destas é descrita de seguida.

A simulação inicia-se pela aplicação das equações de cinemática inversa (equação 4.3) para determinar as velocidades de referência para os motores $v1_{ref}$, $v2_{ref}$, e $v3_{ref}$ a partir das velocidades de referência para o robot v_{ref}, vn_{ref}, e w_{ref}. Estas são depois limitadas e escalonadas segundo o processo descrito em 6.7.1, obtendo-se as velocidades de referência limitadas para cada um dos motores $v1_{limitado}$, $v2_{limitado}$, e $v3_{limitado}$. Utilizando estas velocidades como referência, a simulação dinâmica do comportamento dos motores é efectuada conforme descrito em 6.7.2 de modo a obter as velocidades reais de cada um dos motores $v1_{real}$, $v2_{real}$,e $v3_{real}$. As equações de cinemática directa (equação 4.4) são aplicadas a estas velocidades com o intuito de se obter as velocidades reais em coordenadas do robot v_{real}, vn_{real}, e w_{real}. Finalmente, a equação 4.5 é

aplicada para obter o próximo estado do robot $X_{sim}(k+1)$ a partir do seu estado anterior $X_{sim}(k)$ e das velocidades reais calculadas. Este processo repete-se para cada instante desejado do horizonte de predição, de modo a prever a evolução do sistema durante o horizonte completo.

É habitual que o horizonte de predição N_p seja superior ao horizonte de controlo N_u. Isto significa que o vector de referências de controlo $U_{ref}(k)$ só está definido para os primeiros N_u instantes, enquanto que a predição é feita para N_p instantes. Neste caso são utilizadas como referências de velocidade as definidas em $U_{ref}(k)$ enquanto $k < N_u - 1$. Para $k \geq N_p$ é utilizada a última referência do vector, $U_{ref}(N_u - 1)$.

6.9 Minimização da Função Custo

Sendo o sistema não linear, a função custo tem que ser minimizada recorrendo a métodos numéricos. *Fletcher* descreve em [25] uma série de algoritmos de minimização denominados *métodos de gradiente*. Estes baseiam-se de um modo geral em calcular o gradiente da função que se pretende minimizar, que representa a direcção de maior crescimento da função, e percorrer iterativamente as variáveis independentes na direcção oposta até o mínimo ser encontrado ou o número máximo de iterações ultrapassado. O gradiente de uma função $f(x_1, x_2, ...x_n)$ é definido por:

$$\nabla f(x_1, x_2, ...x_n) = [\frac{df}{dx_1} \frac{df}{dx_2} ... \frac{df}{dx_n}] \tag{6.13}$$

Dos vários métodos descritos por *Fletcher* optou-se por adaptar o denominado *Steepest Descent* aos requisitos do problema de minimização em causa. Duas razões pesaram na escolha deste algoritmo: a sua simplicidade e os bons resultados obtidos por *Scolari* com a sua utilização (ver [1]. O algoritmo, conforme definido em [25], encontra-se descrito no algoritmo 2. Seja $f(x)$ a função a minimizar, $\nabla f(x)$ o gradiente dessa mesma função, e x_{min} o valor de x que a minimiza.

Algorithm 2 Steepest Descent

$k = 0$
$d_0 = -\nabla f(x_0)$
while $d_k < \varepsilon$ **do**
  $x_{k+1} = x_k + \alpha \cdot d_k$
  $k = k + 1$
end while
$x_{min} = x_k$

6.10 Condições de Teste do Controlador

Neste sub-capítulo são descritas as condições de realização dos ensaios para teste e afinação do controlador, assim como as medidas de qualidade definidas para qualificação dos resultados obtidos e o ambiente de simulação utilizado.

6.10.1 Trajectória de Referência

A trajectória de referência para os ensaios do controlador deve apresentar duas características essenciais que tornem os ensaios pertinentes: conter variações bruscas de orientação e de sentido, de modo a representar situações habituais no futebol robótico, e ser seguida a velocidades elevadas, de modo a que o desempenho do controlador seja testado perto dos extremos. Optou-se por criar uma trajectória em "pulso", com orientações de referência diferentes para cada segmento (definida na tabela 6.10.1 e figura 6.8). É de frisar que esta trajectória é constituída por pontos que distam $20cm$ entre eles, tendo os pontos pertencentes ao mesmo segmento um θ_{ref} igual, de modo que a referência de orientação mude bruscamente em cada canto.

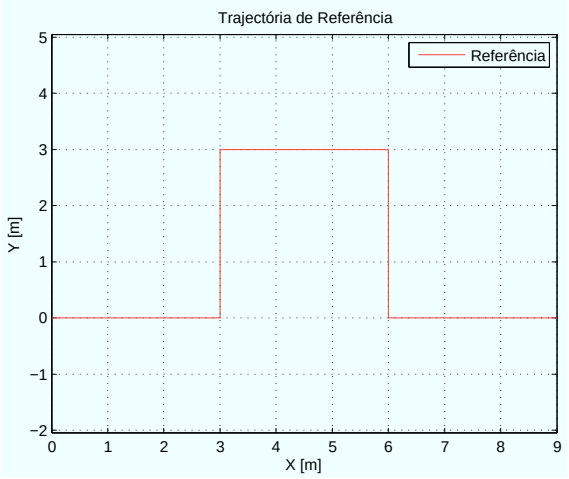


Figura 6.8: Trajectória de referência para os ensaios do controlador

índice (i)	x_{ref} (m)	y_{ref} (m)	$\theta_{ref\ segmento\ i\ a\ i+1}$ (rad)
1	0	0	0
2	3	0	$\pi/2$
3	3	3	0
4	6	3	$-\pi/2$
5	6	0	0
6	9	0	-

Tabela 6.10.1: Pontos que definem a trajectória de referência para ensaios do controlador

6.10.2 Medição do Desempenho do Controlador

De modo a qualificar os resultados obtidos para cada conjunto de parâmetros foram definidas inicialmente quatro medidas de desempenho. São elas:

- **Erro Quadrático Total** $[m^2]$ - Soma dos quadrados das diferenças entre a posição do robot e a posição de referência para cada ponto, medidas na perpendicular à trajectória de referência.
- **Overshoot Máximo e Médio** $[m]$ - Overshoot máximo ocorrido quando o robot faz cada um dos quatro cantos da trajectória e média dos quatro overshoots.
- **Tempo de Estabelecimento Máximo** $[s]$ Tempo máximo que o robot demora a voltar a uma proximidade de $5cm$ da trajectória de referência depois de entrar em overshoot em cada um dos cantos.

Os primeiros ensaios realizados revelaram prontamente que o *Erro Quadrático Total* não era uma medida realista da qualidade dos resultados obtidos, ocorrendo em alguns casos que resultados com overshoot elevado apresentavam um erro quadrático total semelhante a resultados com overshoot muito inferior. Isto deve-se ao facto de um resultado com overshoot mais elevado poder ter um seguimento do restante da trajectória muito preciso, enquanto que o resultado com overshoot menor pode apresentar desvios muito pequenos da trajectória de referência durante todo o percurso. Considera-se portanto como medidas de qualidade principais o *Overshoot Máximo e Médio* e o *Tempo de Estabelecimento Máximo*. Resultados de *Erro Quadrático Total* são ainda assim apresentados como medida da qualidade global do controlador, mas não são considerados nas comparações entre ensaios.

6.10.3 Ambiente de Simulação

Todos os ensaios de simulação foram realizados com o software SimTwo [5]. Este utiliza a biblioteca *Open Dynamics Engine* [43] que garante uma simulação extremamente realista da dinâmica de corpos rígidos, resultando num comportamento dos corpos em simulação muito próximo do comportamento real. Isto, aliado ao facto de os robots utilizados terem sido modelados e rigorosamente parametrizados previamente em SimTwo (ver [44]), faz desta plataforma um ambiente de simulação muito realista e ideal para os ensaios do controlador. Um screenshot do programa pode ser visto na figura 6.10.

Para correr o software duas máquinas numa rede local foram utilizadas, cujo arranjo pode ser visto na figura 6.9. O SimTwo corre no desktop, sobre Windows XP, enquanto que o controlador (mDec) é executado no computador portátil, em Ubuntu 9.10. As duas máquinas comunicam por UDP, enviando o SimTwo ao mDec o estado do robot e da bola e recebendo deste os sinais de controlo do robot. Este arranjo é em tudo igual ao utilizado para controlador robots reais, bastando para isso trocar no mDec o IP do SimTwo pelo ip da máquina do robot real que está a correr o sistema de visão e de interface com o hardware (OVIS).

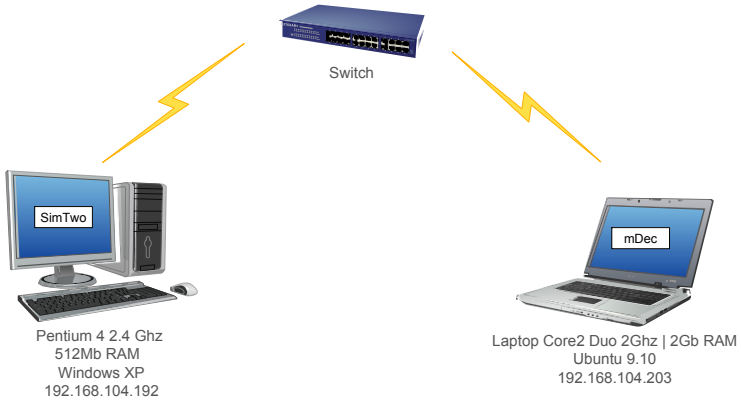


Figura 6.9: Setup de teste em simulação para o Controlador de Trajectória

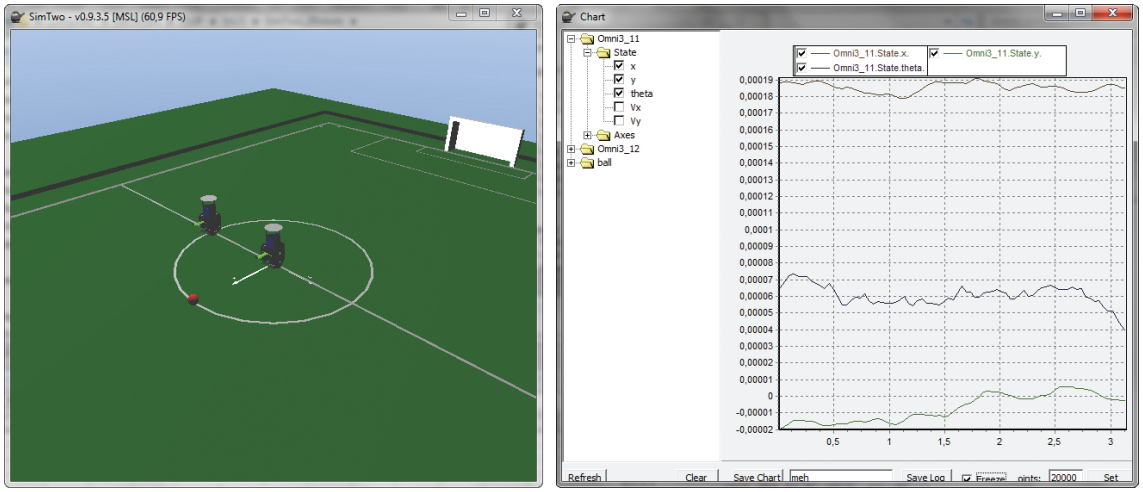


Figura 6.10: SimTwo a correr uma simulação com o modelo do robot utilizado

6.11 Afinação de Parâmetros do Controlador

6.11.1 Afinação de Parâmetros do Optimizador

O algoritmo *Steepest Descent* utilizado para optimizar a função custo, conforme descrito em 6.9, tem apenas um parâmetro configurável: α, o coeficiente de variação das variáveis independentes na direcção de maior decréscimo. Depois de alguns testes iniciais arbitrou-se $\alpha = 0.15$, valor que parece fornecer variações suficientemente pequenas dos sinais de controlo para a minimização da função custo convergir em tempo útil mas não demasiado grandes para que o mínimo seja ultrapassado.

Cada passo de optimização envolve várias simulações de modo a prever a evolução do sistema como resposta a diversas entradas de controlo, razão pela qual o processo de minimização da função custo toma a maior parte do tempo de execução de cada ciclo de controlo. Este ciclo de controlo tem que ser limitado em tempo de modo a não se alargar para além de um intervalo de tempo definido (menos de metade dos $40ms$ disponíveis para cada ciclo de controlo, dado que as tarefas de processamento de imagem para o sistema de visão também são computacionalmente exigentes e têm que correr no mesmo ciclo). Assim, o tempo de execução do algoritmo de minimização tem também ele que ser limitado. Por esta razão definiu-se para o algoritmo um limite máximo de iterações IT_{max} e um critério de paragem ε. A execução do algoritmo pára quando ou o valor da função custo J desce abaixo de ε ou o número de iterações sobe acima de IT_{max}. No segundo caso, as entradas correspondentes ao menor valor da função custo J encontrado até aí são consideradas as óptimas. Estes dois parâmetros foram definidos através da análise dos logs do optimizador, realizando ensaios com $\varepsilon = 0$ e IT_{max} muito elevado.

Verifica-se que para valores de $J \leq 0.1$ a variação das velocidades de referência do robot não é significativa. Executando o algoritmo com um critério de paragem $J = 0.1$ parece fornecer uma minimização suficientemente boa para que o comportamento do robot seja o desejado. Definiu-se então para critério de paragem um valor ligeiramente inferior, $\varepsilon = 0.05$, de modo a garantir alguma tolerância.

Em termos do comportamento do optimizador há dois casos distintos a considerar: quando o robot está a seguir uma recta há já algum tempo e quando está a executar um canto. De seguida apresenta-se o log de uma iteração do algoritmo de controlo no primeiro caso:

```
> Iteration 0 — J: 0.04159994 U: 2.0128 -0.1546 -0.2455
> Iteration 1 — J: 0.04156224 U: 2.0129 -0.1526 -0.2422
> Iteration 2 — J: 0.04152444 U: 2.0130 -0.1507 -0.2389
> Iteration 3 — J: 0.04148652 U: 2.0132 -0.1487 -0.2356
> Iteration 4 — J: 0.04144850 U: 2.0133 -0.1468 -0.2323
> End (stop)
» Time: 5 [ms]
Jbest: 0.04141038
Ref Ubest: 2.0134 -0.1449 -0.2289
Ref vMotor: -1.8607 0.1002 1.6266
```

Porque a referência de velocidade com que é inicializado o algoritmo é a velocidade de referência aplicada ao robot no ciclo de controlo anterior, o algoritmo tem tendência a convergir imediatamente quando o robot se desloca em linha recta durante algum tempo. A primeira iteração do optimizador resulta imediatamente num valor de custo inferior a ε, só sendo executadas cinco iterações do algoritmo de minimização porque este é o limite mínimo de iterações definido. O problema ocorre quando o robot está a executar um canto e o valor da função custo cresce repentinamente devido ao desvio da trajectória de referência. O próximo excerto do log do optimizador corresponde ao início da execução de um canto e demonstra esse efeito:

```
> Iteration 0 — J: 1.76240477 U: 1.9358 0.0675 0.2897
> Iteration 1 — J: 1.76171304 U: 1.9325 0.0691 0.3076
> Iteration 2 — J: 1.76100492 U: 1.9291 0.0708 0.3259
> Iteration 3 — J: 1.76028007 U: 1.9257 0.0725 0.3447
> Iteration 4 — J: 1.75953816 U: 1.9223 0.0741 0.3640
> Iteration 5 — J: 1.75877887 U: 1.9189 0.0758 0.3837
> Iteration 6 — J: 1.75800187 U: 1.9156 0.0774 0.4039
> Iteration 7 — J: 1.75720680 U: 1.9122 0.0790 0.4247
> Iteration 8 — J: 1.75639335 U: 1.9088 0.0807 0.4459
> Iteration 9 — J: 1.75556116 U: 1.9055 0.0823 0.4677
> Iteration 10 — J: 1.75470989 U: 1.9021 0.0839 0.4900
> Iteration 11 — J: 1.75383919 U: 1.8987 0.0855 0.5129
> Iteration 12 — J: 1.75294872 U: 1.8954 0.0871 0.5363
> Iteration 13 — J: 1.75203812 U: 1.8920 0.0887 0.5602
> Iteration 14 — J: 1.75110705 U: 1.8887 0.0903 0.5848
> Iteration 15 — J: 1.75015515 U: 1.8853 0.0919 0.6099
> Iteration 16 — J: 1.74918206 U: 1.8820 0.0935 0.6357
> Iteration 17 — J: 1.74818744 U: 1.8787 0.0950 0.6620
> Iteration 18 — J: 1.74717092 U: 1.8753 0.0966 0.6890
> Iteration 19 — J: 1.74613216 U: 1.8720 0.0982 0.7166
> Iteration 20 — J: 1.74507079 U: 1.8687 0.0997 0.7449
> Iteration 21 — J: 1.74398647 U: 1.8653 0.1012 0.7738
> Iteration 22 — J: 1.74287885 U: 1.8620 0.1028 0.8034
> Iteration 23 — J: 1.74174756 U: 1.8587 0.1043 0.8337
> Iteration 24 — J: 1.74059227 U: 1.8554 0.1058 0.8646
> Iteration 25 — J: 1.73941262 U: 1.8521 0.1073 0.8963
> Iteration 26 — J: 1.73820828 U: 1.8488 0.1088 0.9287
> Iteration 27 — J: 1.73697890 U: 1.8454 0.1103 0.9618
> Iteration 28 — J: 1.73572414 U: 1.8421 0.1118 0.9956
> Iteration 29 — J: 1.73444368 U: 1.8388 0.1133 1.0302
> End (limit)
» Time: 18 [ms]
Jbest: 1.73313719
Ref Ubest: 1.8355 0.1147 1.0656
Ref vMotor: -1.3245 0.0931 1.8548
```

Neste caso a execução do algoritmo terminou porque o número de iterações se encontrava limitado a 30, tendo demorado 18ms. Existem pois casos em que a minimização da função custo não pode ser feita em tempo útil porque requereria um número de iterações exageradamente grande (e consequente tempo de cálculo muito superior ao disponível para cada ciclo de controlo). Existe também possibilidade de, devido às limitações físicas do sistema (que são tidas em conta no modelo sob a forma de limitações de velocidades das rodas e resposta dinâmica dos motores) a função custo simplesmente não poder ser minimizada abaixo do limite inferior definido. Fisicamente, isto significa que dada a posição actual do robot e os limites de velocidade dos motores seria impossível fazer o robot seguir a trajectória desejada à velocidade pretendida durante esse horizonte de predição.

O valor de IT_{max} foi escolhido de maneira a proporcionar o maior número de iterações do algoritmo de minimização possível, mantendo ainda assim o tempo de processamento abaixo de 20ms. Testes com $N_u = 2$ e $N_p = 10$ revelaram que $IT_{max} = 30$ é um bom valor, apresentando o algoritmo de minimização tempos de execução quase sempre inferiores a 20ms para essas 30 iterações. Testes de tempo de execução médio (primeiro valor) e máximo (segundo valor) de tempo de execução do algoritmo de controlo com estes parâmetros são apresentados na tabela 6.11.1, para $8 \leq N_p \leq 12$ e $1 \leq N_u \leq 3$.

Tempos de Execução do Ciclo de Controlo [ms]					
N_u	N_p				
	8	9	10	11	12
1	7.41 16	7.28 15	7.38 16	7.87 17	9.29 18
2	13.10 23	12.280 24	13.30 25	13.60 26	16.77 29
3	18.63 30	18.09 34	17.78 34	21.14 35	23.54 36

Tabela 6.11.1: Tempos de execução médio e máximo do ciclo de controlo Vs. N_p, N_u

Como se pode verificar, para $N_u = 2$ (considerado a base de comparação de testes do algoritmo), este é executado num tempo médio claramente inferior a 20ms. Ocasionalmente, em casos muito excepcionais, o tempo de execução atinge valores superiores a 20ms, mas dada a baixa frequência destes picos não é notório qualquer efeito no desempenho do controlador. Já para $N_u = 3$, onde os tempos médios de execução são consideravelmente mais elevados e os picos acima de 30ms mais frequentes, nota-se um claro efeito no desempenho do controlador. Estes resultados são discutidos em maior detalhe numa secção posterior.

6.11.2 Afinação de Parâmetros do Modelo

Além do tempo reduzido de simulação, uma outra vantagem de utilizar um modelo simplificado é a diminuição do número de parâmetros do mesmo que se tem que afinar. No caso do

modelo utilizado são dois: o valor de velocidade limite dos motores, para o algoritmo de saturação e escalonamento, e a constante de tempo do sistema que modela a resposta dos motores ao degrau.

6.11.2.1 Limite de Velocidade dos Motores

O limite de velocidade angular das rodas, tendo em conta o motor e a caixa redutora de 12:1, foi obtido por simulação em SimTwo e fixa-se em 633 RPM (rotações por minuto). Isto corresponde a $\frac{633}{60} * 2 * \pi = 66.3\ rad/s$, o que para um raio da roda de $0.051cm$ se traduz numa velocidade linear de $3.38\ m/s$. Estes valores são consistentes com a datasheet dos motores utilizados (Maxon RE-40 com caixa redutora GP0 42C), que indicam uma velocidade máxima sem carga de $\frac{7580}{12} = 632RPM$, o que corresponde a $66.2rad/s$ e portanto $3.37m/s$. Testes realizados com o robot real indicam, no entanto, que os motores reais não são capazes de atingir esta velocidade.

O algoritmo IRV utilizado para modelar o limite de velocidade das rodas está no entanto a ser utilizado de um modo ligeiramente mais criativo de maneira a limitar artificialmente a velocidade dos motores quando o robot está a executar um canto com rotação. Quando é pedido ao robot que execute uma trajectória com um determinado valor de velocidade linear de referência, a velocidade máxima que pode ser requerida de uma determinada roda ocorre quando apenas duas das três rodas estão a realizar o esforço de movimento. O caso mais óbvio em que o movimento é unicamente assegurado por duas das rodas ocorre quando a direcção da velocidade de referência é a mesma da velocidade linear do robot (o vector v da figura 4.1). Das equações de cinemática apresentadas na equação 4.3 conclui-se que neste caso a velocidade das duas rodas motrizes será $sin(\pi/3) * v$.

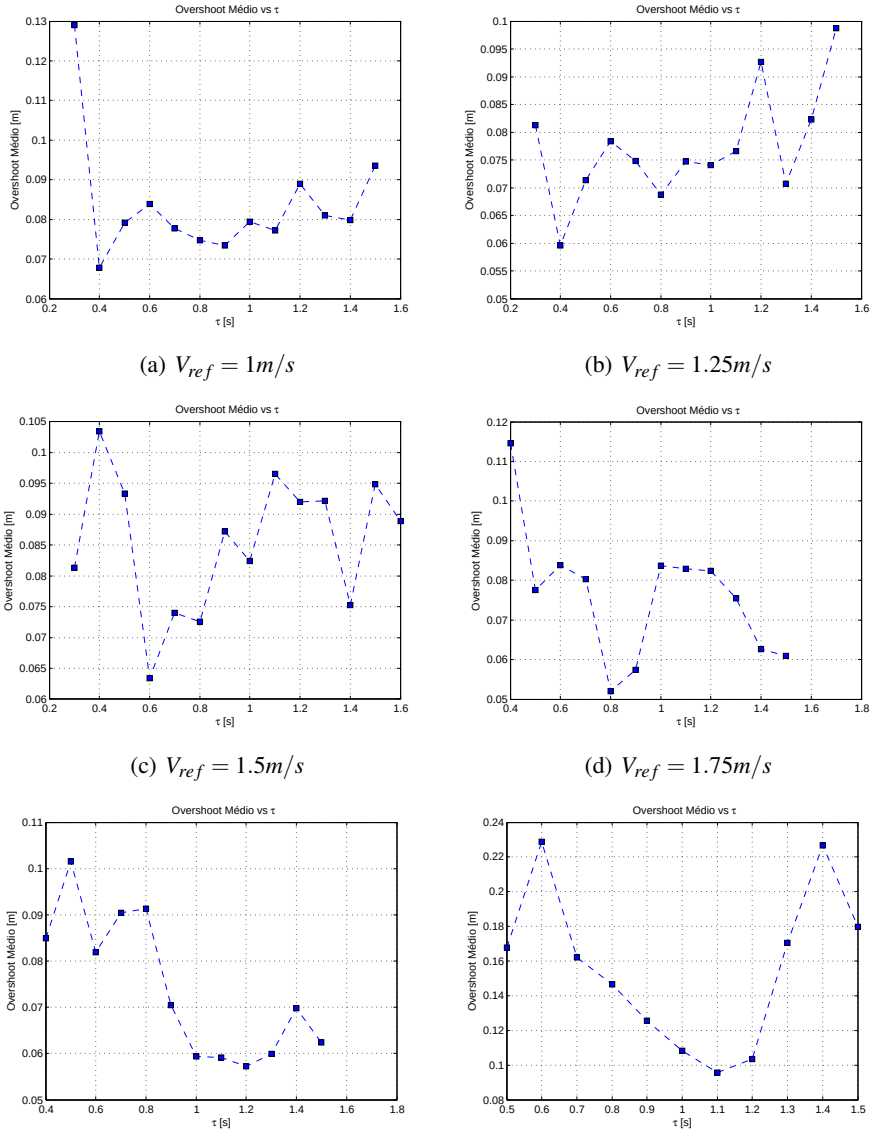
O limite de velocidade angular das rodas é então alterado dinamicamente durante a execução do algoritmo para um valor igual a $sin(\pi/3) * v_{ref}$. Isto permite que o robot atinja a velocidade pretendida em termos de velocidade linear ao mesmo tempo que limita o valor máximo da velocidade dos motores quando o robot entra em rotação, evitando-se velocidades de rotação excessivas que perturbam o seguimento da trajectória.

6.11.2.2 Constante de Tempo do Sistema

A constante de tempo do sistema refere-se à constante de tempo do sistema LTI de primeira ordem com o qual se aproximou a resposta do conjunto de motor + controlador de velocidade existente no robot para cada roda. Um sistema LTI de primeira ordem tem uma resposta invariante ao degrau. Isto significa que independentemente do valor do degrau e do estado actual do sistema, este vai sempre demorar τ segundos a atingir $1 - 1/e$ (isto é, aproximadamente 63.2%) do valor final. Isso não acontece no sistema motor + controlador de velocidade real, onde o sistema evolui tanto mais rapidamente para as velocidades desejadas quanto mais baixas são as velocidades em jogo ou mais pequena a variação das mesmas. Isto deve-se ao efeito de saturação da tensão (duty cycle do PWM) aplicada aos motores. Para velocidades baixas o controlador PI utilizado tem uma grande margem de tensão a aplicar aos motores, resultando em acelerações rápidas e uma constante de tempo baixa. Para velocidades altas o controlador está já a operar perto dos limites, fornecendo aos motores uma tensão próxima da tensão de saturação e portanto sem grande margem

para acelerações bruscas. Isto leva a grandes períodos de saturação da tensão no valor máximo e consequentemente a uma resposta mais lenta. Por esta razão a constante de tempo τ do modelo tem que ser adequada às velocidades de referência pretendidas. Esta variação da constante de tempo modeliza no fundo o efeito de saturação do PWM descrito atrás.

Executaram-se então uma série de ensaios às velocidades lineares de referência de 1, 1.25, 1.5, 1.75, 2.0, e 2.25 m/s variando em cada ensaio τ entre os valores pertinentes. Para cada valor de τ realizou-se uma simulação com a trajectória atrás definida e obtiveram-se os valores de *Overshoot Máximo e Médio* para cada, considerando-se esta a principal medida de qualidade. Destes ensaios, cujos gráficos de Overshoot Médio são apresentados na figura 6.11, procurou-se obter a constante de tempo ideal para cada valor de velocidade de referência.



(a) $V_{ref} = 1m/s$ (b) $V_{ref} = 1.25m/s$

(c) $V_{ref} = 1.5m/s$ (d) $V_{ref} = 1.75m/s$

(e) $V_{ref} = 2m/s$ (f) $V_{ref} = 2.25m/s$

Figura 6.11: Ensaios para determinação de τ (overshoot médio Vs. τ)

A análise dos gráficos da figura 6.11 revela o esperado: de um modo geral, quanto mais altas forem as velocidades de referência em jogo maior é o valor da constante de tempo mais adequada ao sistema, variando entre $0.4s$ e $1.3s$. É também de notar que, para cada ensaio, os valores de overshoot médio para as constantes de tempo próximas da constante de tempo que resulta em overshoot mínimo são muito próximos, variando por vezes menos de 2 cm. Dada a aplicação em causa estas diferenças não são significativas. Especialmente para as velocidades mais elevadas ($V_{ref} = 2m/s$ e $V_{ref} = 2.25m/s$), um considerável intervalo de τ resulta num overshoot médio semelhante (1 a 1.3 s para $V_{ref} = 2m/s$ e 1 a 1.2 s para $V_{ref} = 2.5m/s$). Pretendendo-se determinar uma relação matemática entre o τ ideal e V_{ref} é importante ter isto em conta. A figura 6.12 apresenta as zonas dos melhores valores de τ para cada V_{ref} utilizado nos ensaios.

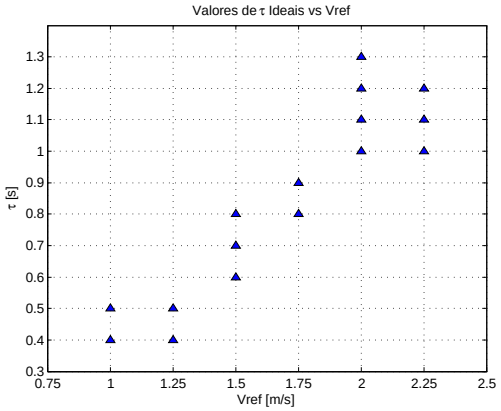


Figura 6.12: Possíves valores de τ ideais para cada V_{ref}

A observação da distribuição dos pontos do gráfico da figura 6.12 sugere que talvez seja possível adaptar com bons resultados a relação τ - V_{ref} a uma recta. A recta da equação 6.14, visível na figura 6.13 adapta-se convenientemente aos pontos da figura.

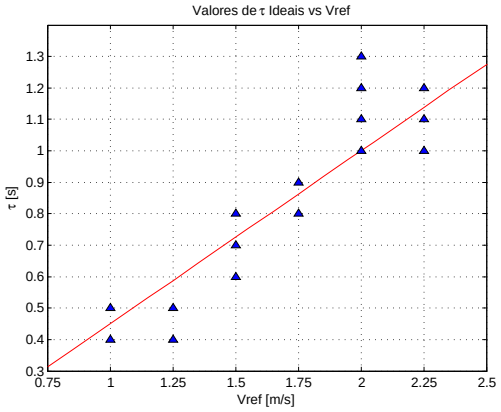
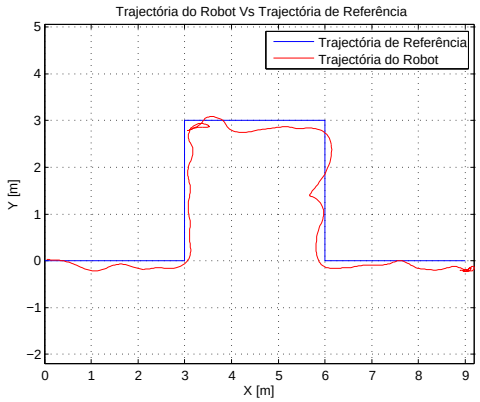


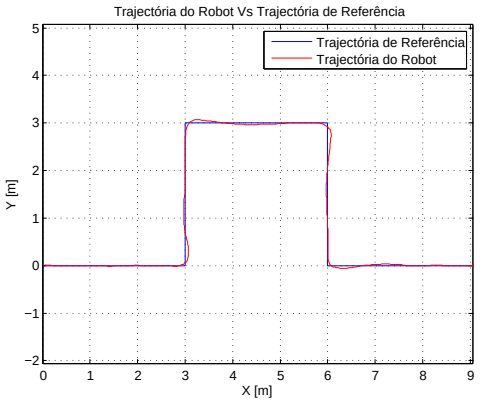
Figura 6.13: Relação τ - V_{ref} aproximada por uma linha recta

$$\tau = 0.55 * V_{ref} - 0.1 \tag{6.14}$$

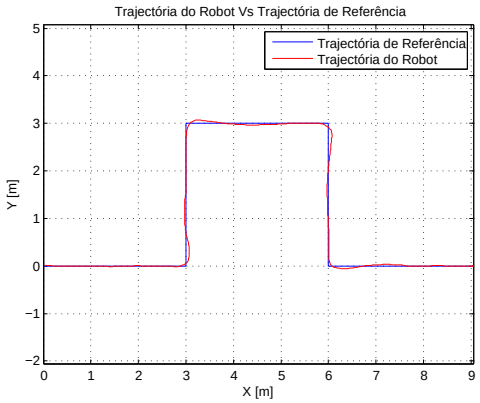
Há no entanto um efeito importante a considerar: para valores muito pequenos de τ o modelo apresenta uma resposta praticamente instantânea a uma referência de velocidade, o que significa que se aproxima muito do modelo puramente cinemático. Ora para horizontes de controlo N_u superiores a 1 o controlador só apresenta um bom desempenho se um modelo dinâmico for utilizado. Compare-se na figura 6.14 a diferença nas trajectórias seguidas pelo robot com $\tau = 0.3$ e $\tau = 0.4$. O desempenho extremamente pobre do controlador com $\tau = 0.3s$ deve-se não à inadequação da constante de tempo à velocidade de referência mas ao efeito referido acima. Nos casos em que a constante de tempo é calculada como sendo inferior a $0.4s$ (velocidades de referência inferiores a $0.91m/s$, segundo a equação 6.14) é preferível comutar os parâmetros do controlador para $N_u = 1$ e um modelo puramente cinemático. Nesse caso obtém-se uma trajectória bastante mais aceitável do que utilizando $N_u \geq 2$ e um modelo dinâmico com $\tau < 0.4$ (ver gráfico da figura 6.14).



(a) Modelo Dinâmico ($N_u = 2$, $N_p = 10$, e $\tau = 0.3s$)



(b) Modelo Dinâmico ($N_u = 2$, $N_p = 10$, e $\tau = 0.4s$)



(c) Modelo Puramente Cinemático ($N_u = 1$ e $N_p = 10$)

Figura 6.14: Ensaios do controlador, $V_{ref} = 1m/s$

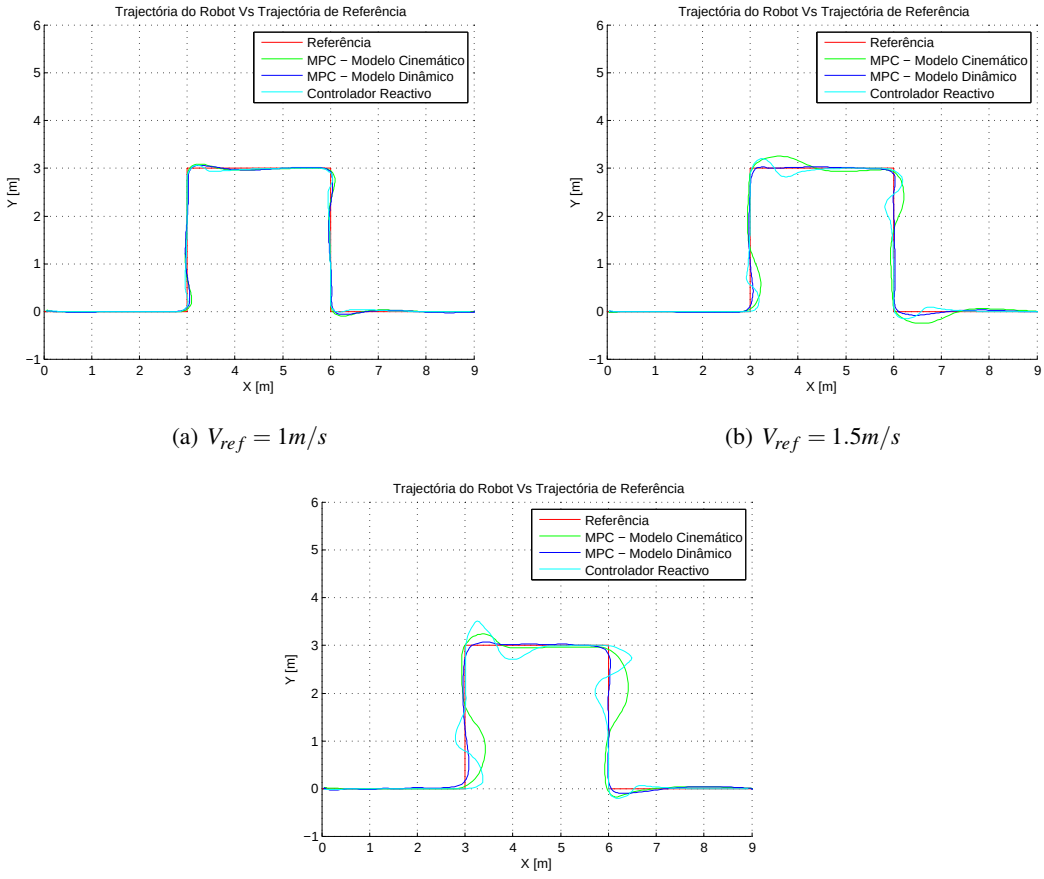
6.12 Ensaios do Controlador em Simulação

Os ensaios a seguir descritos foram realizados com o intuito de documentar o desempenho do controlador para várias configurações possíveis. São apresentados resultados que comparam a utilização de um modelo puramente cinemático com o modelo dinâmico, resultados para diferentes valores de N_u e N_p, e desempenho para trajectórias onde a referência de orientação é fixa ou varia suavemente.

6.12.1 Efeito da Utilização de Diferentes Modelos

É importante comparar o desempenho do controlador preditivo aqui projectado com o mesmo controlador utilizando um modelo puramente cinemático e com um controlador reactivo. O controlador reactivo testado é o actualmente utilizado nos robots da equipa 5DPO e encontra-se descrito em [1].

Para cada controlador realizaram-se ensaios a velocidades de referência de $1m/s$, $1.5m/s$, e $2m/s$. Os resultados são apresentados na figura 6.15.



(a) $V_{ref} = 1m/s$

(b) $V_{ref} = 1.5m/s$

(c) $V_{ref} = 2.0m/s$

Figura 6.15: Ensaios para comparação do desempenho de diferentes controladores

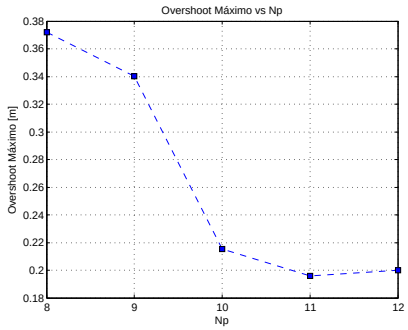
Dos resultados apresentados na figura 6.15 inferem-se algumas conclusões importantes. A primeira é a grande vantagem, nesta aplicação, dos controladores preditivos face aos reactivos. O controlador reactivo aqui testado, embora apresentando um desempenho aceitável a baixas velocidades, demonstra uma degradação considerável de qualidade com o aumento da velocidade. Isto é bem visível nos ensaios a $1.5m/s$ e $2m/s$, onde a trajectória efectuada apresenta acentuados overshoots e undershoots. Já ambos os controladores preditivos apresentam um desempenho claramente mais consistente em todo o alcance de velocidades de referência testado, com overshoots sempre inferiores e trajectórias bem mais suaves. Isto advém da sua capacidade de anteciparem as curvas, começando a adequar as velocidades dos motores à execução deste antes de o robot o encontrar.

A outra é a importância da utilização de um bom modelo dinâmico para o robot. Este é particularmente importante a velocidades elevadas, já que a $1m/s$ ambos os controladores com modelo puramente cinemático e com modelo dinâmico simplificado apresentam um desempenho semelhante. No entanto, a velocidades elevadas, as trajectórias correspondentes ao controlador com modelo puramente cinemático apresentam overshoots consideráveis enquanto que as do controlador com modelo dinâmico simplificado são consistentemente melhores, com overshoots bem mais reduzidos. A correcta simulação da dinâmica é então tanto mais importante quanto maiores forem as velocidades em jogo. Dos resultados obtidos retira-se também que a simplificação utilizada na modelação dinâmica do robot parece ser suficientemente próxima da realidade para fornecer resultados de qualidade dentro do alcance de velocidade considerado. Estes ensaios sugerem também que a baixas velocidades (abaixo de $1m/s$) pode ser vantajoso comutar os parâmetros do controlador para $N_u = 1$ e utilizar um modelo puramente cinemático, pois o peso computacional do controlador fica consideravelmente reduzido.

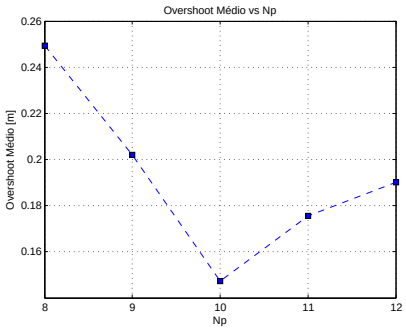
6.12.2 Efeito dos Horizontes de Predição e Controlo

Nesta secção são apresentados resultados dos ensaios para determinação do efeito da alteração dos parâmetros N_u e N_p (horizontes de controlo e predição, respectivamente) no desempenho do controlador. Realizaram-se para este efeito ensaios a $V_{ref} = 2m/s$, com $1 \leq N_u \leq 3$ e $8 \leq N_p \leq 12$.

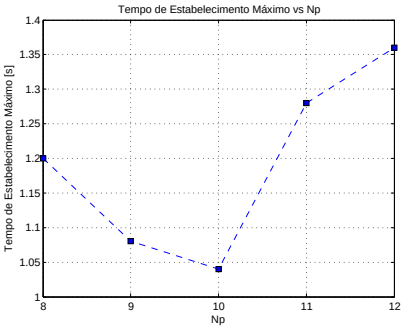
Os valores de overshoot médio e máximo, tempo de estabelecimento, e de erro total quadrático são apresentados sob a forma de gráficos para cada combinação de N_u e N_p e compilados depois em tabelas para melhor comparação. Assim, a figura 6.16 contém os valores das medidas de desempenho obtidos para $N_u = 1$ e $8 \leq N_p \leq 12$. As figuras 6.17 e 6.18 apresentam o mesmo mas para $N_u = 2$ e $N_u = 3$, respectivamente. Os valores exactos destas grandezas encontram-se reproduzidos nas tabelas 6.12.1, 6.12.2, 6.12.3, e 6.12.4



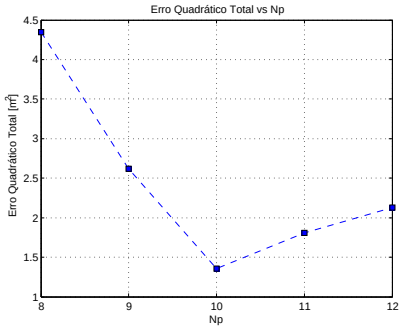
(a) Overshoot Máximo



(b) Overshoot Médio

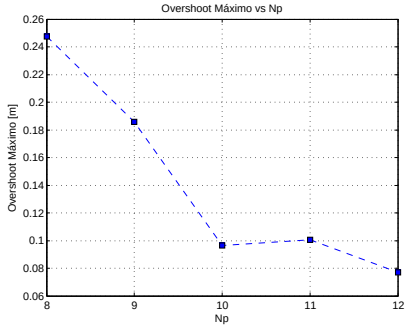


(c) Tempo de Estabelecimento Máximo

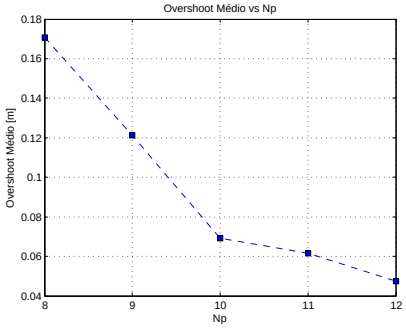


(d) Erro Total Quadrático

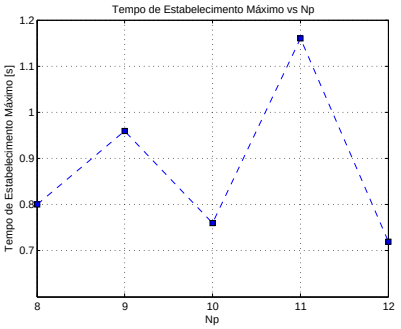
Figura 6.16: Medidas de desempenho para $N_u = 1$ e $8 \leq N_p \leq 12$



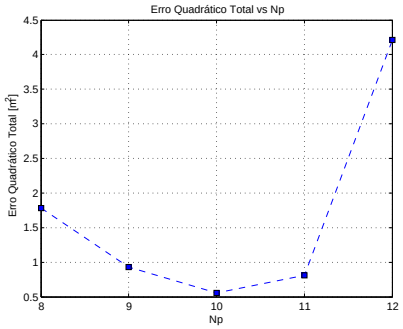
(a) Overshoot Máximo



(b) Overshoot Médio

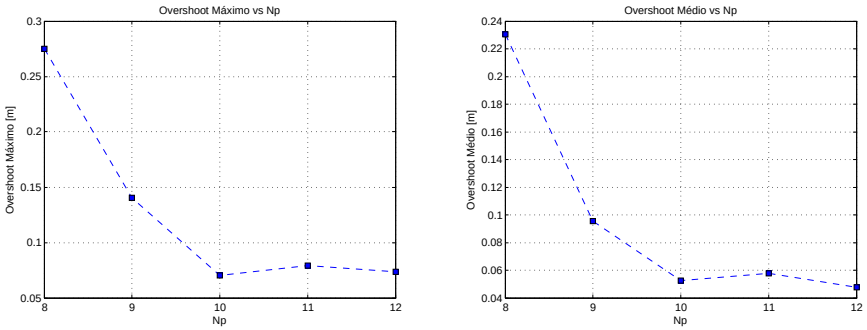


(c) Tempo de Estabelecimento Máximo



(d) Erro Total Quadrático

Figura 6.17: Medidas de desempenho para $N_u = 2$ e $8 \leq N_p \leq 12$



(a) Overshoot Máximo

(b) Overshoot Médio

(c) Tempo de Estabelecimento Máximo

(d) Erro Total Quadrático

Figura 6.18: Medidas de desempenho para $N_u = 3$ e $8 \leq N_p \leq 12$

Overshoot Máximo [m] Vs N_p, N_u					
N_u	N_p				
	8	9	10	11	12
1	0.3720	0.3400	0.2153	0.1958	0.2000
2	0.2474	0.1857	0.0966	0.1006	0.0770
3	0.2749	0.1406	0.0703	0.0791	0.0733

Tabela 6.12.1: Valores de Overshoot Máximo

Overshoot Médio [m] Vs N_p, N_u					
N_u	N_p				
	8	9	10	11	12
1	0.2493	0.2019	0.1472	0.1754	0.1901
2	0.1707	0.1212	0.0692	0.0618	0.0475
3	0.2305	0.0954	0.0528	0.0579	0.0478

Tabela 6.12.2: Valores de Overshoot Médio

Tempo de Estabelecimento Máximo [s] Vs N_p, N_u					
N_u	N_p				
	8	9	10	11	12
1	1.2000	1.0800	1.0400	1.2800	1.3600
2	0.8000	0.9600	0.7600	1.1600	0.7200
3	1.0400	0.6800	1.2400	1.1200	0.9200

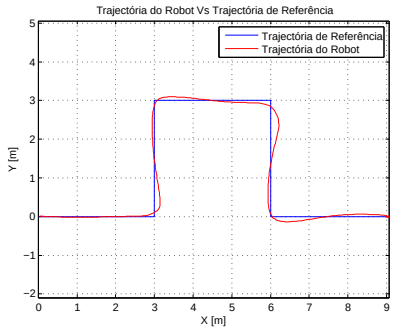
Tabela 6.12.3: Tempo de Estabelecimento Máximo

Erro Total Quadrático [m^2] Vs N_p, N_u					
N_u	N_p				
	8	9	10	11	12
1	4.3482	2.6187	1.3574	1.8096	2.1301
2	1.7828	0.9327	0.5535	0.8102	4.2139
3	3.7554	0.5596	0.4597	3.0645	0.9721

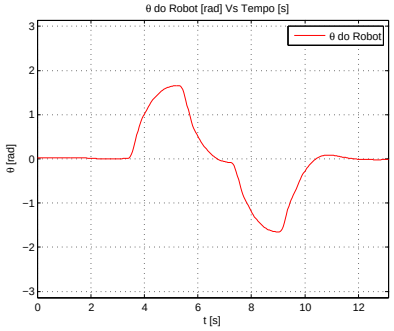
Tabela 6.12.4: Erro Total Quadrático

Analise-se em primeiro lugar o efeito de N_u no desempenho do controlador. A análise das tabelas demonstra claramente que, para um mesmo valor de N_p, a qualidade dos resultados melhora com o aumento de N_u. Os valores de overshoot médio e máximo, principais medidas de desempenho, descem consideravelmente com o incremento de N_u de 1 para 2, decrescendo igualmente com o incremento de 2 para 3 (se bem que consideravelmente menos). A diferença de desempenho é óbvia e claramente visível nos gráficos das trajectórias apresentados na figura 6.19. Há no entanto um efeito importante a considerar: devido ao modo como o algoritmo de minimização da função custo funciona (nomeadamente, ao modo como o vector gradiente da função custo é calculado) cada incremento unitário de N_u resulta num aumento considerável do tempo de processamento do algoritmo (claramente demonstrado na tabela 6.11.1). O ganho em desempenho ao incrementar N_u de 2 para 3 não compensa o tempo acrescido de computação do algoritmo, que

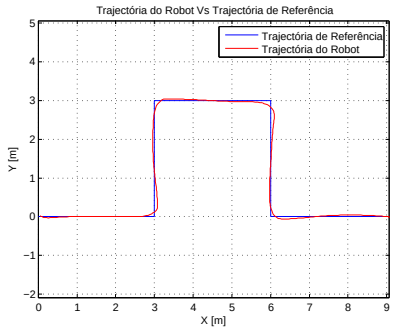
pode causar perdas de sincronismo (afectando o desempenho do controlador) ou afectar as tarefas de processamento de imagem. $N_u = 2$ apresenta-se então como o valor mais adequado para o horizonte de controlo.



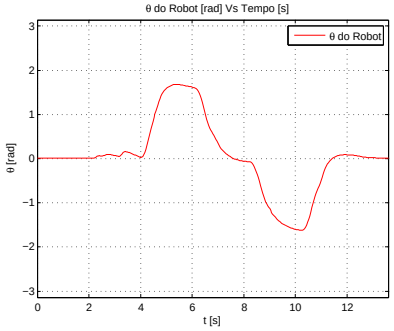
(a) $N_u = 1, N_p = 10$ (XY)



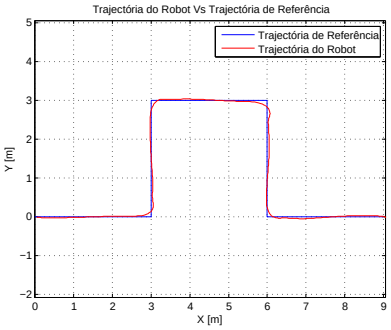
(b) $N_u = 1, N_p = 10$ (θ)



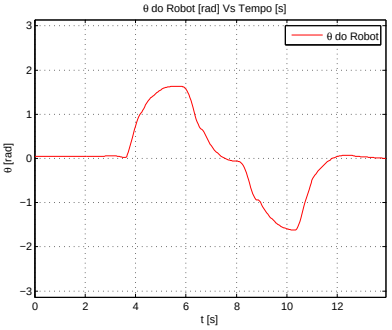
(c) $N_u = 2, N_p = 10$ (XY)



(d) $N_u = 2, N_p = 10$ (θ)



(e) $N_u = 3, N_p = 10$ (XY)

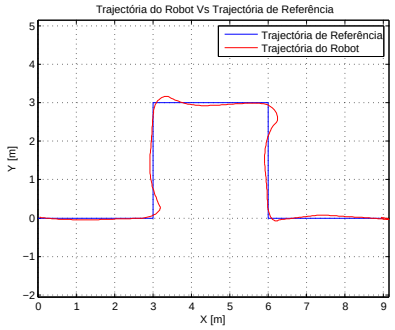


(f) $N_u = 3, N_p = 10$ (θ)

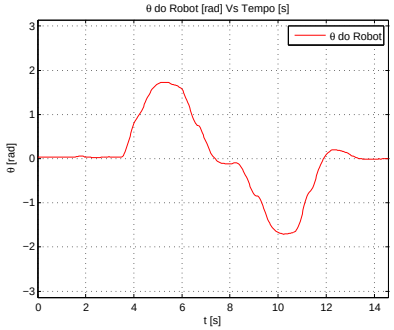
Figura 6.19: Comparação de resultados com variação de N_u

O efeito da alteração de N_p é ao mesmo tempo mais subtil e mais complexo. Conforme esperado, o tempo de execução do algoritmo (ver tabela 6.11.1) cresce com o aumento de N_p, não apresentando no entanto um aumento tão drástico como o provocado pelo incremento de N_u. Mais uma vez, isto deve-se ao modo de cálculo da função custo. Também à semelhança do incremento de N_u, os valores de overshoot máximo e médio decrescem com o aumento de N_p, deixando adivinhar uma melhoria do desempenho. Atente-se, no entanto, nos resultados apresentados na figura 6.20. Para $N_p = 8$ ocorre um overshoot considerável, explicado pelo facto de o controlador

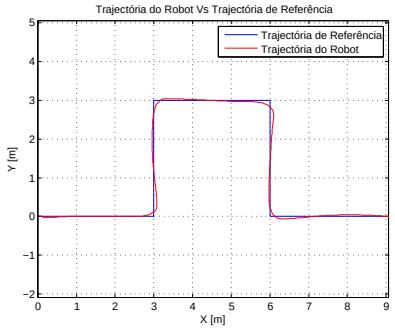
não incorporar na predição informação suficientemente futura e começar a executar o canto tarde demais, o que às velocidades elevadas em jogo resulta num desvio da trajectória. Com $N_p = 12$ o contrário ocorre: o controlador prevê o canto cedo demais e executa a curva demasiado "por dentro", falhando os pontos de canto da trajectória. Em suma, horizontes de predição demasiado pequenos resultam em overshoots grandes e horizontes demasiado grandes no corte exagerado de cantos. Os resultados obtidos para $N_p = 10$ parecem apresentar o melhor equilíbrio possível entre estes dois efeitos, onde a trajectória do robot sofre algum overshoot mas passa mais perto dos cantos.



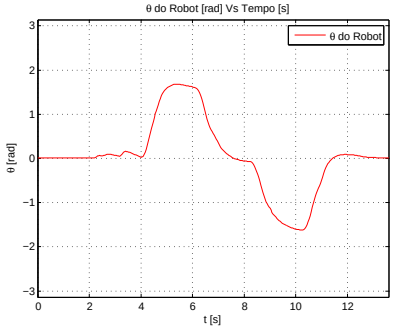
(a) $N_u = 2, N_p = 8$ (XY)



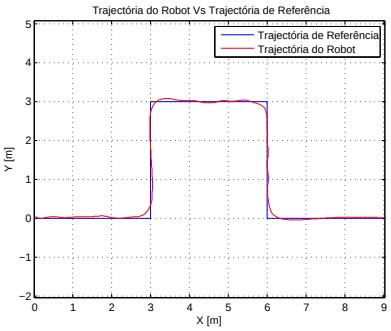
(b) $N_u = 2, N_p = 8$ (θ)



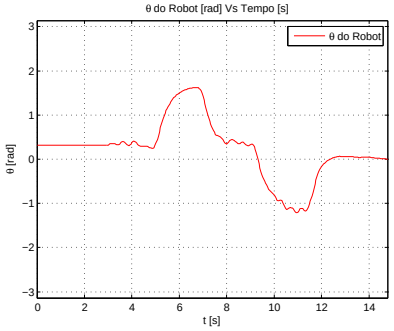
(c) $N_u = 2, N_p = 10$ (XY)



(d) $N_u = 2, N_p = 10$ (θ)



(e) $N_u = 2, N_p = 12$ (XY)



(f) $N_u = 2, N_p = 12$ (θ)

Figura 6.20: Comparação de resultados com variação de N_p

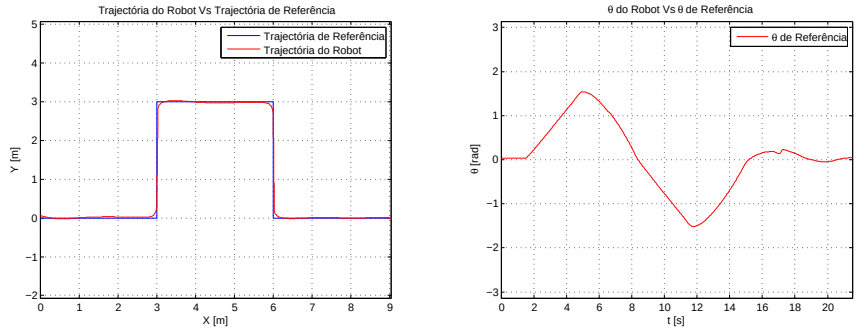
6.12.3 Efeito das Variações de θ

De modo a documentar o comportamento do controlador para casos em que a variação de θ_{ref} é suave em vez de brusca criou-se uma trajectória de teste definida por apenas 6 pontos (ao invés da utilizada até agora, definida por pontos espaçados de $20cm$). Isto permite que o controlador faça uma interpolação suave de θ enquanto percorre cada segmento, em vez de trocar bruscamente a orientação em cada canto. Esta trajectória encontra-se definida na tabela 6.12.5.

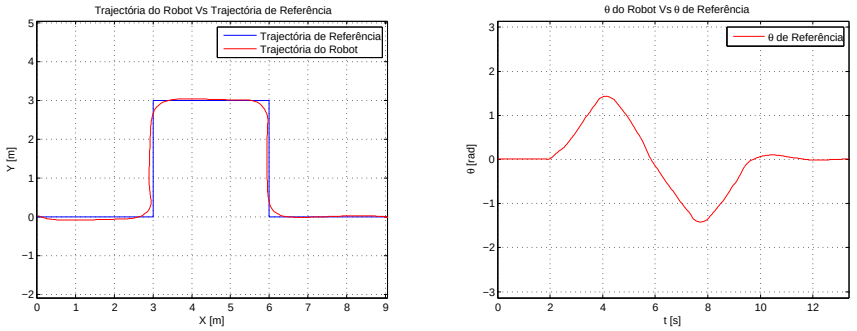
índice (i)	x_{ref} (m)	y_{ref} (m)	θ_{ref} (rad)
1	0	0	0
2	3	0	$\pi/2$
3	3	3	0
4	6	3	$-\pi/2$
5	6	0	0
6	9	0	0

Tabela 6.12.5: Trajectória de referência com variação de θ_{ref} suave

Foram realizados ensaios com esta trajectória a $V_{ref} = 1m/s$ e $V_{ref} = 2m/s$. Os resultados encontram-se apresentados na figura 6.21.



(a) $V_{ref} = 1m/s$ (XY) (b) $V_{ref} = 1m/s$ (θ)

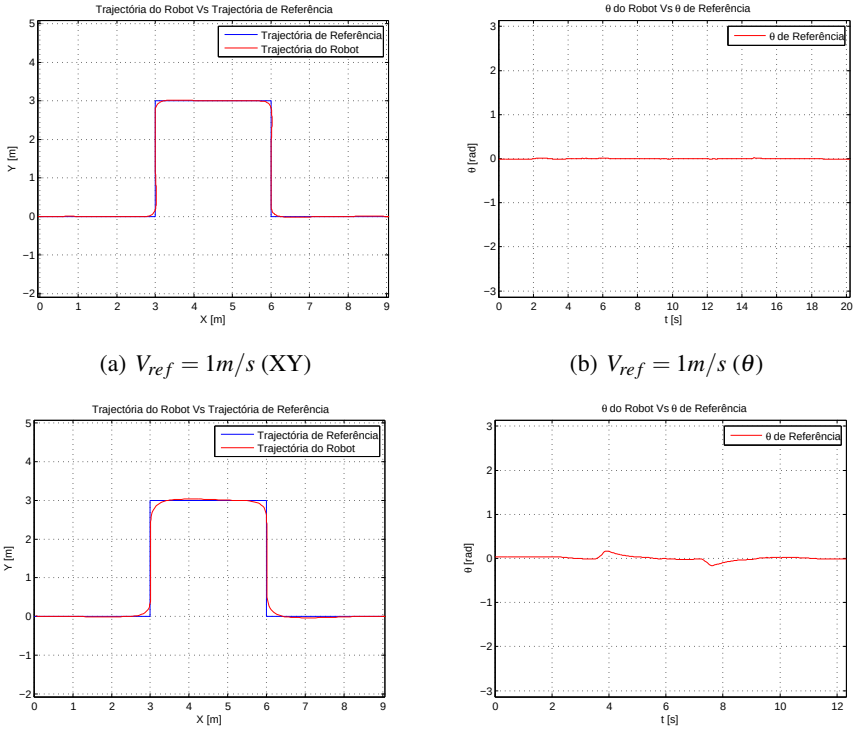


(c) $V_{ref} = 2m/s$ (XY) (d) $V_{ref} = 2m/s$ (θ)

Figura 6.21: Ensaios com variação de θ_{ref} suave

Os resultados mostram que se durante a trajectória de referência θ_{ref} variar suavemente (veja-se os gráficos de θ da figura 6.21) o desempenho do controlador é significativamente melhor. Para $V_{ref} = 1m/s$ a trajectória é seguida com muita precisão, sem overshoot significativo. Com $V_{ref} = 2m/s$ o robot afasta-se ligeiramente da trajectória depois do primeiro canto mas apresenta ainda assim overshoot muito diminuto. O desempenho do controlador é claramente melhor quando a trajectória contém variações de orientação suaves, ao longo de segmentos com algum tamanho, do que quando a referência é subitamente comutada.

Foram ainda realizados ensaios em que a referência de orientação é fixa ($\theta_{ref} = 0$) em toda a trajectória. Os resultados, para $V_{ref} = 1m/s$ e $V_{ref} = 2m/s$, são apresentados na figura 6.22.



(a) $V_{ref} = 1m/s$ (XY)

(b) $V_{ref} = 1m/s$ (θ)

(c) $V_{ref} = 2m/s$ (XY)

(d) $V_{ref} = 2m/s$ (θ)

Figura 6.22: Ensaios com $\theta_{ref} = 0$ para toda a trajectória

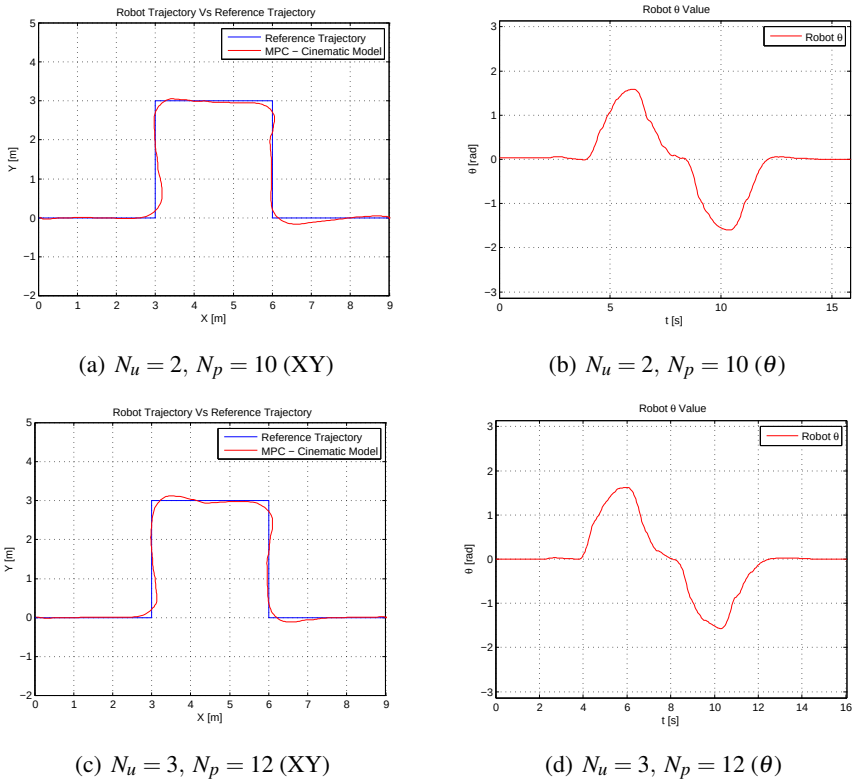
No caso de uma referência de orientação fixa o desempenho é ainda melhor. A trajectória percorrida a $V_{ref} = 1m/s$ é praticamente idêntica à trajectória de referência, sem overshoots discerníveis. A $V_{ref} = 2m/s$ os segmentos são seguidos muito de perto e não existe overshoot significativo, mas os cantos são realizados mais "por dentro"do que a $V_{ref} = 1m/s$. Isto explica-se pelo modo como é calculada a trajectória de referência para o controlador tendo em conta a velocidade (descrito na secção 6.6), já que a velocidades maiores uma maior quantidade da trajectória vai ser tida em conta para o mesmo período de controlo e o robot começa a virar mais cedo.

Em suma, em ambos os casos (de θ_{ref} fixo ou variações θ_{ref} suaves) o desempenho do controlador é notável e consideravelmente melhor do que para trajectórias contendo variações bruscas de referência de orientação.

6.12.4 Efeito da Escolha do Optimizador

Tomou-se o facto de ter sido implementado um novo algoritmo de optimização para utilização no controlador de formações (RPROP, descrito na secção 7.6) como uma oportunidade para fazer alguns ensaios posteriores de modo a determinar a influência do optimizador nos resultados finais. Pretendia-se determinar especificamente se o algoritmo era capaz de convergir mais depressa que o *Steepest Descent* na situação descrita na secção 6.11.1.

Assim, realizaram-se dois testes semelhantes aos anteriores, a uma velocidade de referência de $2m/s$, e com horizontes de controlo de 2 e 3 e de predição de 10 e 12. Os resultados encontram-se apresentados na figura 6.23.



(a) $N_u = 2$, $N_p = 10$ (XY)

(b) $N_u = 2$, $N_p = 10$ (θ)

(c) $N_u = 3$, $N_p = 12$ (XY)

(d) $N_u = 3$, $N_p = 12$ (θ)

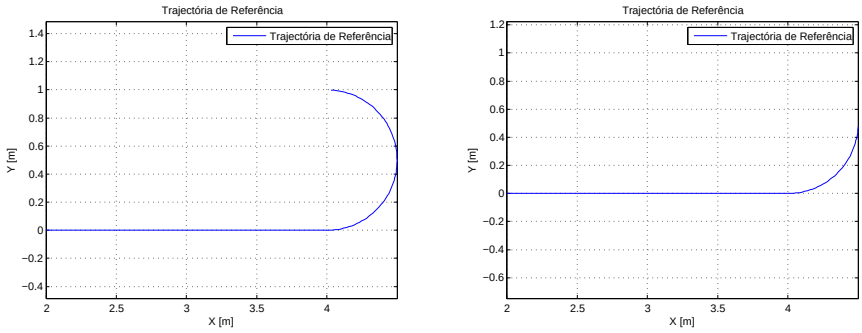
Figura 6.23: Ensaios utilizando o algoritmo de optimização RPROP, $v_{ref} = 2m/s$ e θ_{ref} variável

O efeito da troca do algoritmo de optimização é claramente visível quando comparando os resultados obtidos com os apresentados na secção 6.12.2. De um modo geral o robot aparenta começar a virar mais cedo, realizando os cantos mais "por dentro". Dado que os horizontes de predição e de controlo são iguais aos ensaios realizados com *Steepest Descent* isto pode ser explicado pelas características diferentes da optimização realizado pelo RPROP. No entanto isto não se traduziu em overshoots máximos mais reduzidos, pelo que as vantagens da utilização deste algoritmo para seguimento de trajectórias fixas são duvidosas. Para mais, o RPROP parece fornecer resultados piores quando se trata do seguimento de rectas. A análise dos logs do optimizador também não revelou grandes diferenças na convergência da optimização, chegando este algoritmo a resultados de custo finais semelhantes para situações semelhantes.

No entanto, os resultados aqui obtidos para o RPROP são algo falaciosos. São-no na medida em que os resultados apresentados para o *SteepestDescent* são fruto de uma enorme quantidade de ensaios e consequente optimização de parâmetros enquanto que os do RPROP foram ensaios breves realizados rapidamente antes do prazo de finalização do projecto. Os parâmetros do optimizador não foram minuciosamente adequados ao problema em questão, e como tal estes resultados não são uma comparação perfeitamente imparcial dos dois algoritmos.

6.12.5 Trajectória em Gancho

Finalmente realizaram-se alguns ensaios com uma trajectória de aplicação directa no futebol robótico habitualmente denominada trajectória em gancho. Nesta o robot segue em linha recta até à bola e realiza um semi-círculo ou quarto de círculo em torno dela, de modo a posicionar-se rapidamente para um remate. Assim, a trajectória de referência corresponde a uma recta com referência de orientação variável (entre a orientação inicial do robot e a orientação final que ele deve manter para ficar de frente para a bola) seguida de um semi-círculo ou de um quarto de círculo com 0.5 m de raio, onde a referência de orientação é fixa. Estas trajectórias de referência podem ser vistas na figura 6.24.

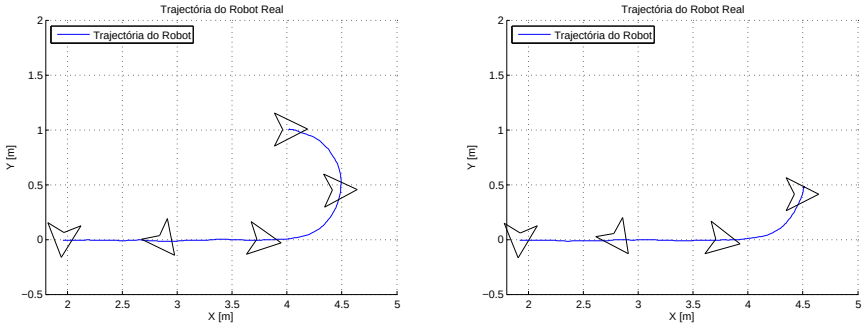


(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.24: Trajectórias de Referência em Gancho

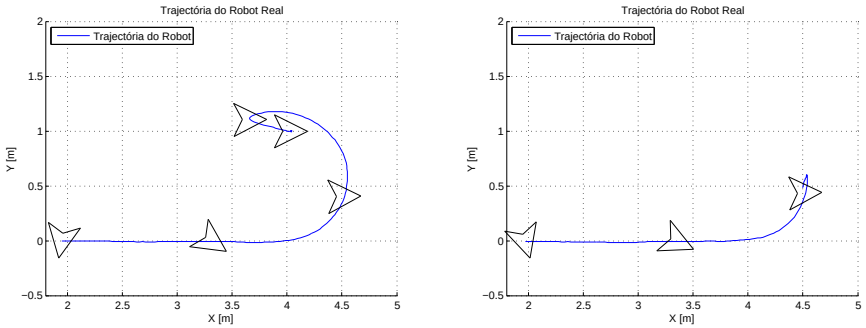
Os resultados, a velocidades de 0.5 m/s, 0.75 m/s e 1 m/s são apresentados de seguida. Por uma questão de facilidade de visualização e poupança de espaço, sobre os plots X/Y foi marcado um símbolo correspondente ao estado robot a intervalos de 2 segundos. A parte côncava do triângulo corresponde à parte frontal do robot. Da sua posição e orientação pode ser retirado o estado do robot nesse instante.

Empregando o controlador preditivo o robot é capaz de executar a trajectória desejada a $0.5m/s$ sem falhar o ponto final, posicionando-se correctamente para rematar a bola (figura 6.25). Com a subida da velocidade para $0.75m/s$ (figura 6.26) os arcos executados pelo robot já são mais largos do que os desejados e o robot falha o ponto final por uma distância considerável, tendo que voltar atrás. O mesmo acontece para uma velocidade de $1m/s$ (figura 6.27), com um desvio ainda maior do arco de referência. No caso do arco em quarto de circunferência isto não se torna muito preocupante porque o desvio não ocorre na direcção da bola e o robot é capaz de se posicionar



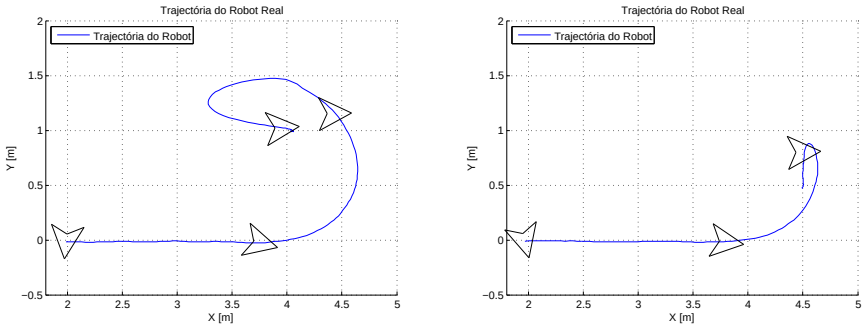
(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.25: Trajectória em Gancho, $V_{ref} = 0.5m/s$



(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.26: Trajectória em Gancho, $V_{ref} = 0.75m/s$

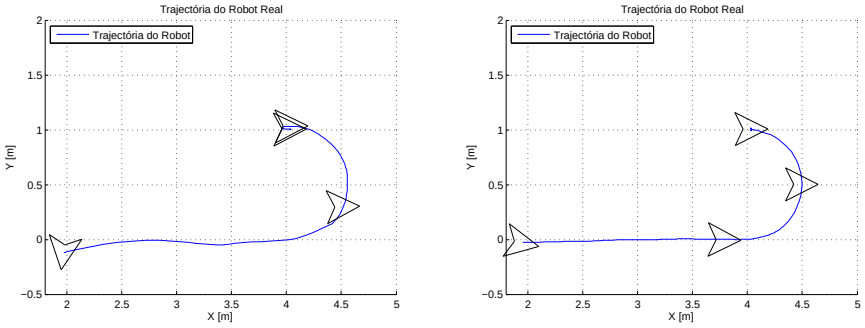


(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.27: Trajectória em Gancho, $V_{ref} = 1.0m/s$

correctamente sem colidir com ela. Já no caso no semi-círculo o robot embateria mesmo na bola, o que obviamente é indesejável.

Os resultados aqui obtidos demonstram que o ideal é ter para a mesma trajectória duas velocidades de referência: durante a secção recta pode ser utilizada uma velocidade elevada, devendo esta referência ser diminuída para $0.5m/s$ para a execução do arco. Isto pode ser conseguido de duas formas: ou a referência de velocidade é abruptamente trocada imediatamente antes do início



(a) Comutação abrupta de V_{ref} (b) Variação suave de V_{ref}

Figura 6.28: Trajectória em Gancho com variação de V_{ref}

do arco ou suavemente diminuída ao longo da recta de maneira a atingir o arco já com a velocidade de $0.5m/s$. As duas situações estão ilustradas na figura 6.28, com uma velocidade na recta de $2m/s$ e no arco de $0.5m/s$. Conforme esperado, o robot desloca-se com velocidade elevada durante a recta desacelerando para executar o arco, conseguindo assim posicionar-se correctamente sem desvios.

6.13 Ensaios do Controlador em Situação Real

Após os ensaios em simulação o algoritmo de controlo desenvolvido foi também testado no robot real com vista a validar os resultados obtidos. Devido ao método de obtenção de dados dos ensaios (descrito na secção abaixo) não foi possível registar dados do robot real a executar uma trajectória de teste idêntica à das simulações, tendo-se registado apenas a execução de um canto (equivalente ao primeiro canto da trajectória de teste completa).

Há também a considerar que os resultados obtidos para os ensaios reais sofrem de uma fonte de erro muito considerável e inexistente nas simulações: o erro de localização. Enquanto que o robot da simulação tem sempre a sua localização exacta, o robot real posiciona-se recorrendo a um misto de odometria e localização baseada em visão artificial. Nestes testes apenas a odometria foi utilizada, sendo que a localização baseada em visão provoca demasiada oscilação na posição estimada do robot, o que é pouco desejável no seguimento de trajectórias. Ora os erros de odometria acumulam-se e são tanto piores quanto maior for a velocidade do robot, pelo que será importante ter isto em conta na análise dos resultados.

6.13.1 Obtenção de Dados

Os dados de posição e orientação do robot durante os ensaios foram obtidos adaptando o sistema de visão utilizado pelos robots da liga Small da RoboCup, SmallVis. Este consiste numa única câmara montada sobre o campo e apontada para baixo, com o plano de visão perfeitamente paralelo ao chão. Cada robot possui um marcador circular no topo, que o software SmallVis é capaz de reconhecer e assim determinar a posição e orientação do robot. Para fornecer medições

viáveis este sistema requer uma calibração precisa tanto da câmara como do software. Como referência, este processo será descrito de seguida.

Calibração da Câmara - A câmara foi montada numa calha no tecto sobre o campo onde foram executados os ensaios a 2.87 metros do chão, abrangendo uma área de cerca de 3 m por 2.5 m. Este distância foi medida desde o chão até ao ponto médio entre o extremo da objectiva e a posição do sensor, uma vez que a existência de múltiplos pontos focais ao longo da objectiva não permite saber a localização exacta de nenhum deles. De modo a reduzir o efeito de vinhetagem a abertura da objectiva foi reduzida até ao valor mínimo que permitisse obter uma imagem que o software fosse capaz de processar de modo a identificar os marcadores.

De modo a garantir que o plano da imagem fosse perfeitamente paralelo ao chão utilizou-se um fio de prumo alinhado com o centro da objectiva para marcar no chão o ponto que ficava no alinhamento vertical exacto desta. Nesse ponto marcou-se uma cruz e verificou-se na imagem produzida pela câmara se esta correspondia ao centro exacto do frame. Fizeram-se então pequenos ajustes na posição da câmara de modo a que isto acontecesse e marcou-se um novo ponto da mesma maneira. O processo foi repetido ciclicamente até que o ponto marcado coincidisse exactamente com o centro da imagem produzida pela câmara.

Calibração do Software - A utilização do SmallVis requer a calibração prévia de três elementos: as cores do marcador, o efeito de distorção em barril da objectiva, e o sistema de coordenadas.

De modo a calibrar o sistema de coordenadas considerou-se o ponto central marcado anteriormente como a origem do referencial $(0,0)$. Marcaram-se em torno deste 11 cruzes separadas por $0.5m$ no eixo x e $1m$ no eixo y, para um total de 12 pontos de calibração (figura 6.29). No software adicionaram-se pontos sobre as cruzes marcadas no campo apresentadas na imagem e atribuíram-se-lhes as coordenadas correspondentes aos pontos reais.

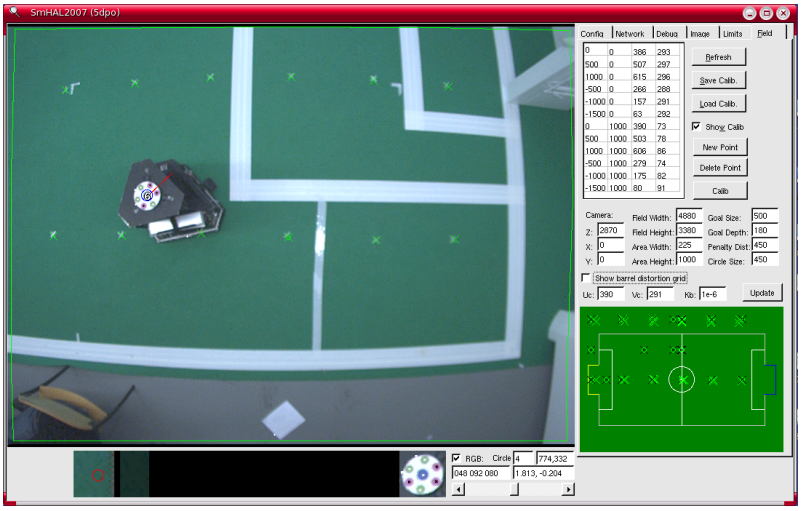


Figura 6.29: SmallVis, calibração do sistema de coordenadas e detecção do marcador

A calibração do efeito de distorção em barril consistiu em variar os parâmetros da grelha que corrige este efeito até que as linhas desta se adequassem às linhas do campo apresentadas na imagem. Este efeito pode ser visto na figura 6.30.

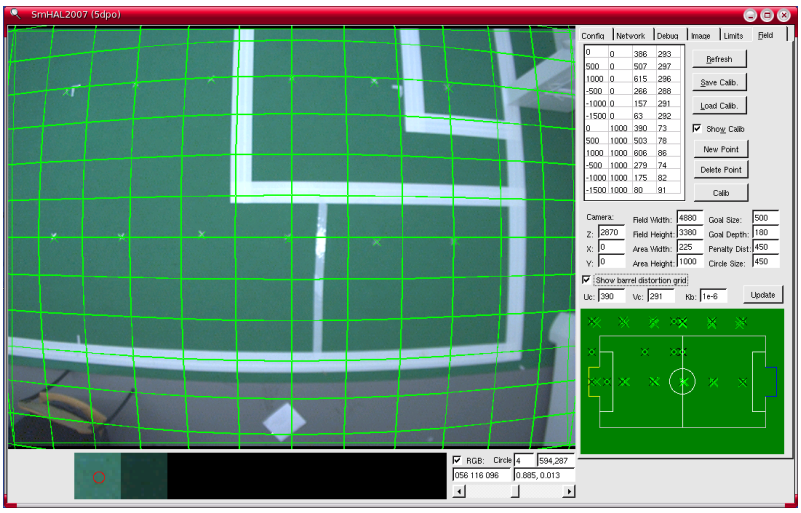


Figura 6.30: SmallVis, calibração do efeito de distorção em barril

A calibração do marcador consiste em obter amostras das cores do mesmo vistas pela câmara e fazê-las corresponder às cores que o software procura. O marcador utilizado encontra-se reproduzido na figura 6.31.

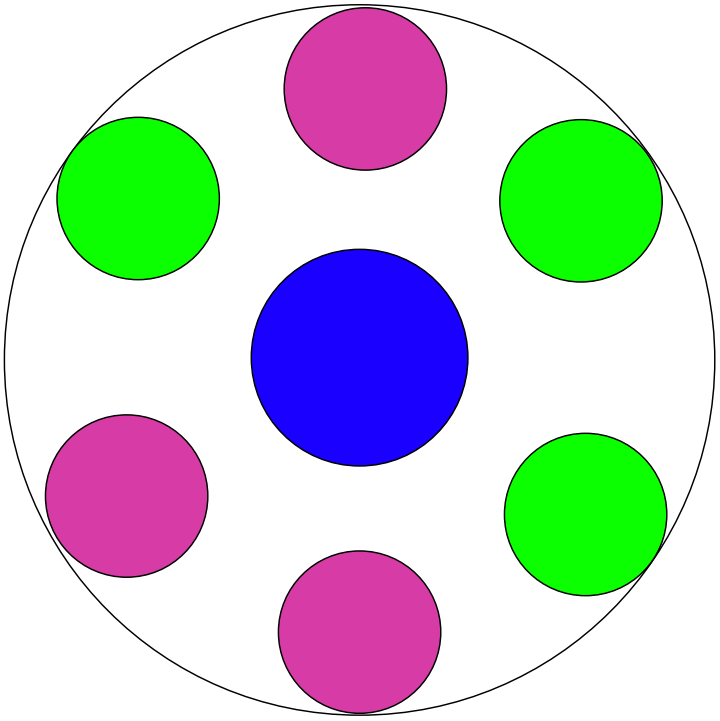


Figura 6.31: Marcador utilizado pelo SmallVis para localização do robot

As cores presentes no marcador não correspondem exactamente às descritas no software, pelo que a cor "*white*"do software deve ser calibrada para o púrpura do marcador (deve-se estritamente a razões históricas, devido à altura em que esta era a cor utilizada na liga pequena de futebol robótico). O "blue"e "green"do software correspondem exactamente ao azul e verde do marcador. Esta calibração deve ser executada em várias zonas do campo, dado que a luz raramente é uniforme ao longo da imagem completa e as cores são percepcionadas de maneira diferente pelo software em áreas com iluminação diferente. Uma vez calibradas as cores correctamente o marcador deve ser identificado prontamente e um número deve aparecer por cima da sua posição na imagem (como pode ser visto na figura 6.29).

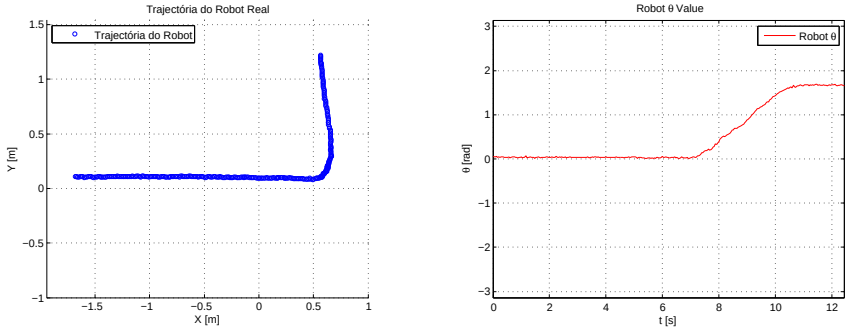
6.13.2 Resultados Obtidos

Foram realizados ensaios fazendo o robot executar um canto em ângulo recto com variação brusca de orientação de 0 para $\pi/2$ radianos (equivalente ao primeiro canto da trajectória de referência utilizada nos testes em simulação). Testaram-se velocidades de referência entre $0.5m/s$ e $2m/s$, utilizando o controlador MPC projectado. O controlador reactivo utilizado anteriormente nos robots foi também testado sucintamente, de maneira a obter-se uma base de comparação.

Porque o erro de localização do robot, especialmente a velocidades elevadas, tem uma forte influência nos erros que ocorrem durante o seguimento da trajectória, torna-se difícil discernir exactamente se estes são derivados do controlador ou do erro de localização. Os resultados obtidos são no entanto suficientes para atestar o funcionamento do controlador no robot real e realizar algumas comparações entre os algoritmos reactivo e preditivo. O SmallVis foi utilizado para produzir um log da posição e orientação do robot para cada ensaio, sendo os resultados obtidos apresentados como plots de posição X/Y e θ em função do tempo.

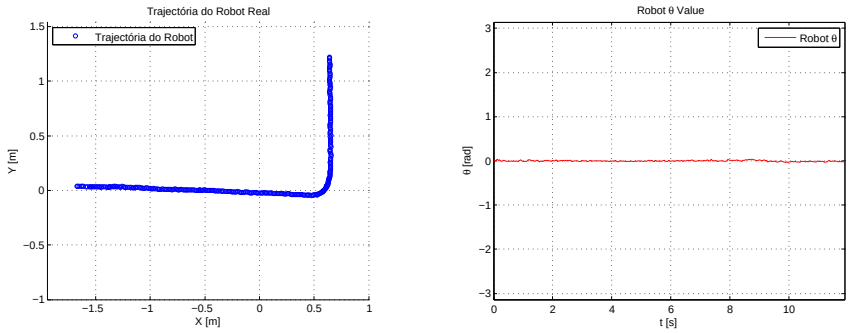
6.13.2.1 Controlador Preditivo, $V_{ref} = 0.5m/s$

De seguida apresentam-se os resultados dos ensaios realizados utilizando o controlador MPC para $V_{ref} = 0.5m/s$. A figura 6.32 corresponde ao ensaio com variação de θ_{ref} brusca no canto e a figura 6.33 ao ensaio em que a referência de orientação é fixa e igual a 0.



(a) Plot XY (b) Plot θ

Figura 6.32: Ensaio com variação de θ_{ref} brusca e $V_{ref} = 0.5m/s$



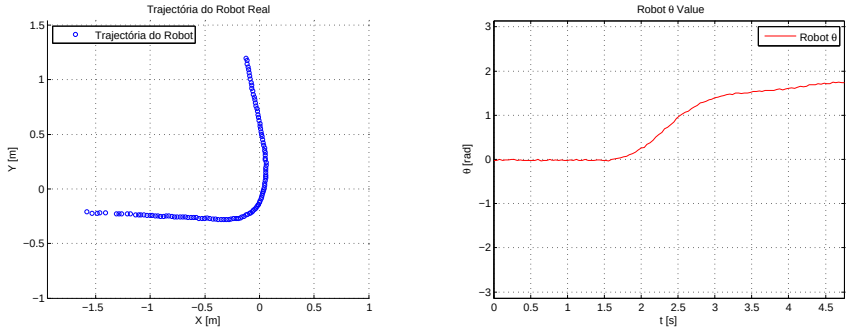
(a) Plot XY (b) Plot θ

Figura 6.33: Ensaio com $\theta_{ref} = 0$ e $V_{ref} = 0.5m/s$

Em ambos os casos é notório o efeito preditivo do controlador. Ao aproximar-se do canto o robot desacelera e começa a fazer a curva por dentro, resultando na execução suave do mesmo e num overshoot reduzido. O caso em que existe uma variação brusca da orientação apresenta um overshoot moderado no canto, resultado consistente com os resultados obtidos em simulação. Dada a baixa velocidade do robot os erros de odometria são consideravelmente minimizados e o robot é capaz de se localizar correctamente e seguir as rectas definidas na trajectória de referência.

6.13.2.2 Controlador Preditivo, $V_{ref} = 1m/s$

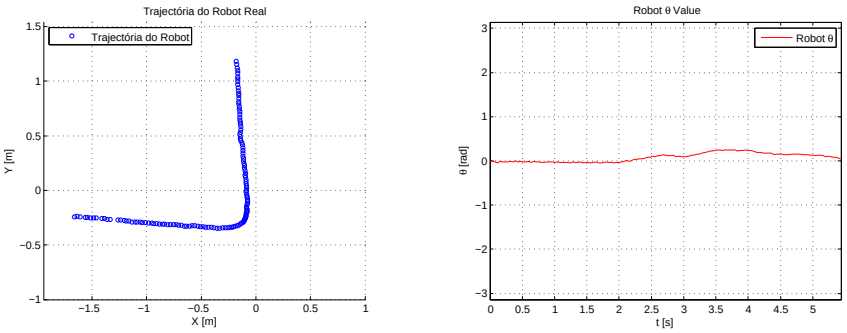
As figuras seguintes correspondem aos resultados dos ensaios realizados com $V_{ref} = 1.0m/s$ (figura 6.34 para o ensaio com variação de θ_{ref} brusca no canto e figura 6.35 para o ensaio em que a referência de orientação é fixa e igual a 0).



(a) Plot XY (b) Plot θ

Figura 6.34: Ensaio com variação de θ_{ref} brusca e $V_{ref} = 1.0m/s$

Aqui começam a ser visíveis os erros de localização devido à odometria, com a recta inicial a ser seguida ligeiramente oblíqua ao invés de perfeitamente horizontal (enquanto que o robot reportava estar a progredir na direcção correcta). Se se considerar este primeiro troço como horizontal, desprezando de certo modo os efeitos do erro de localização, os resultados são ainda bastante interessantes: a trajectória com referência de orientação fixa apresenta um overshoot mínimo e a com variação de referência brusca apresenta um valor de overshoot moderado.

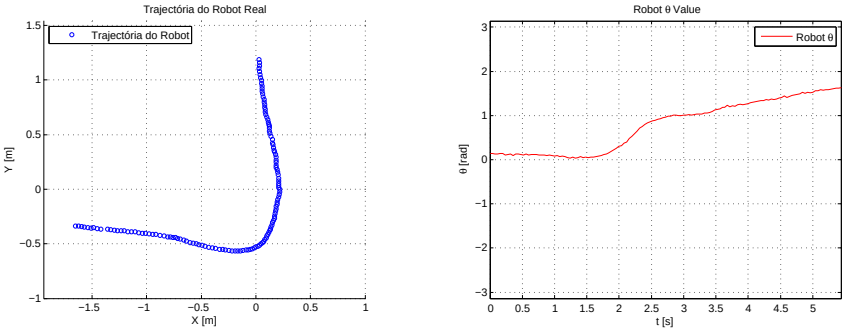


(a) Plot XY) (b) Plot θ

Figura 6.35: Ensaio com $\theta_{ref} = 0$ e $V_{ref} = 1.0m/s$

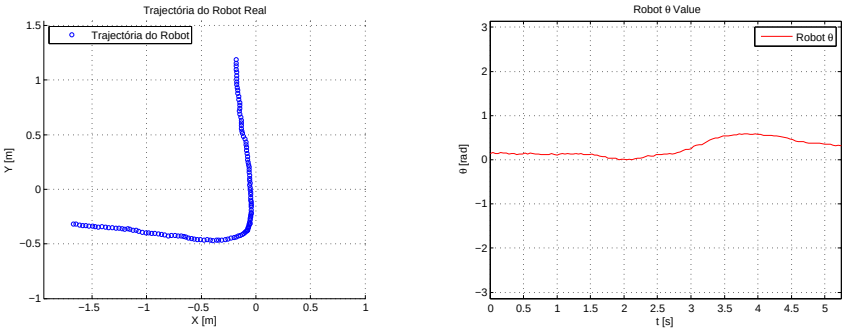
6.13.2.3 Controlador Preditivo, $V_{ref} = 1.5m/s$

De seguida apresentam-se os resultados obtidos para uma referência de velocidade de $1.5m/s$. A figura 6.36 corresponte aos ensaios com variação de θ_{ref} brusca no canto e a figura 6.37 aos resultados do ensaio em que a referência de orientação é fixa e igual a 0.



(a) Plot XY (b) Plot θ

Figura 6.36: Ensaio com variação de θ_{ref} brusca e $V_{ref} = 1.5m/s$



(a) Plot XY (b) Plot θ

Figura 6.37: Ensaio com $\theta_{ref} = 0$ e $V_{ref} = 1.5m/s$

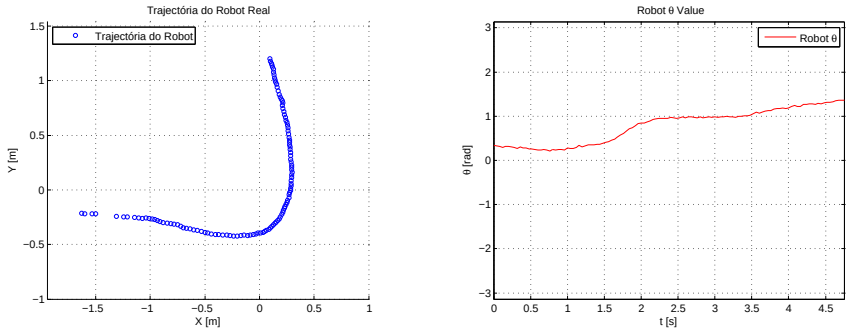
Com o incremento de velocidade os resultados começam a ficar claramente piores, crescendo o valor de overshoot nos dois casos. É no entanto difícil discernir quanta desta degradação de

desempenho é causada por erros de localização e quanta é culpa directa do controlador, dado que o primeiro segmento de recta é seguido já com algum erro.

6.13.2.4 Controlador Preditivo, $V_{ref} = 2m/s$

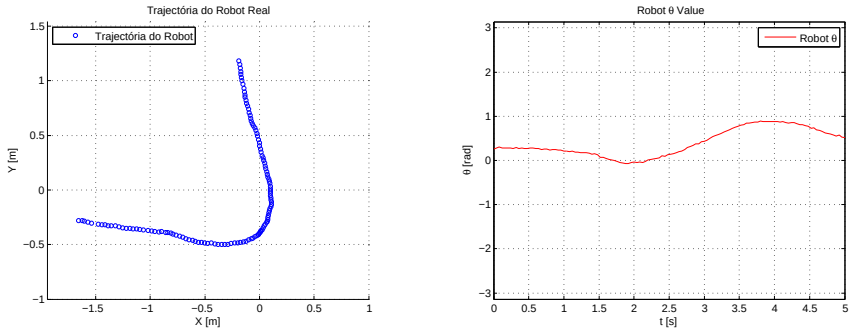
Finalmente são apresentados os resultados dos ensaios efectuados com uma velocidade de referência de $2m/s$, expostos nas figuras 6.38 e 6.39. Com o aumento da velocidade de referência a degradação da qualidade dos resultados observada na situação anterior é ainda mais acentuada, mas mais uma vez é difícil dizer quanto diz respeito ao erro de localização do robot devido à odometria e quanto depende do próprio controlador.

Embora o robot fosse capaz de atingir velocidades superiores a $2m/s$ o seu movimento era demasiado errático para ser possível garantir a integridade física do mesmo, uma vez que os ensaios foram realizados num local próximo de paredes, cadeiras, e da baliza do campo de futebol. Como tal, optou-se por não se fazer ensaios a velocidades superiores a $2m/s$.



(a) Plot XY (b) Plot θ

Figura 6.38: Ensaio com variação de θ_{ref} brusca e $V_{ref} = 2.0m/s$



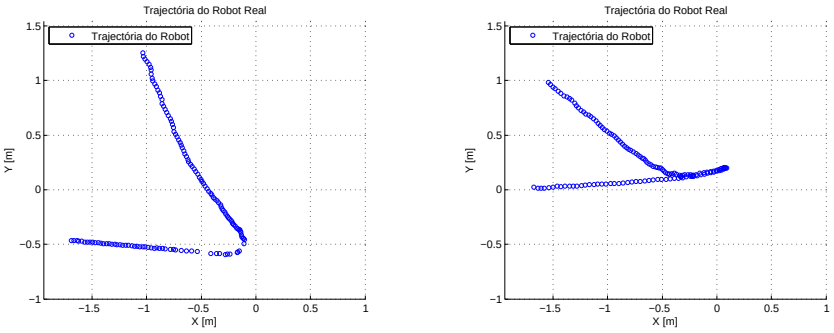
(a) Plot XY) (b) Plot θ

Figura 6.39: Ensaio com $\theta_{ref} = 0$ e $V_{ref} = 2.0m/s$

6.13.2.5 Controlador Reactivo

Foram também realizados alguns ensaios com o controlador reactivo testado previamente em simulação, utilizando uma trajectória de teste idêntica à aplicada nos testes imediatamente anteri-

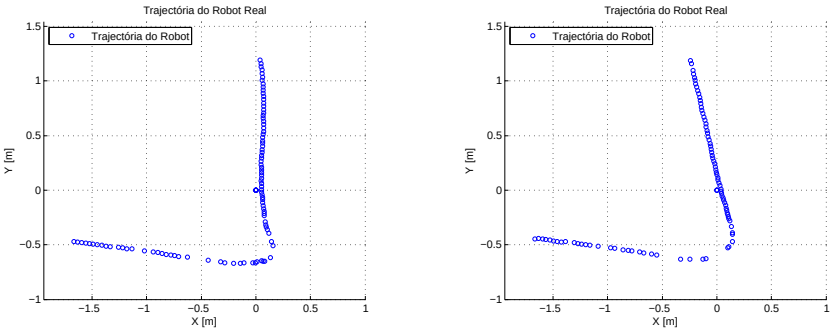
ores. A figura 6.40 apresenta os resultados obtidos para a trajectória com $\theta_{ref} = 0$, a velocidades de $1m/s$ e $1.5m/s$.



(a) $V_{ref} = 1m/s$ (b) $V_{ref} = 1.5m/s$

Figura 6.40: Ensaios com controlador reactivo, $\theta_{ref} = 0$

Com referência de orientação fixa os resultados obtidos foram claramente fracos, especialmente para velocidades iguais ou superiores a $1m/s$. Ocorre nestes casos que o robot, ao chegar ao canto, trava bruscamente ao invés de desacelerar e começar a virar como o controlador preditivo. Durante esta travagem a transferência de massas resulta na inclinação do robot para a frente, com a roda traseira a ficar a girar no ar (um efeito semelhante ao ocorrido em motas ou bicicletas ao travar com o travão frontal). Este facto confunde completamente a localização baseada em odometria, e resulta num seguimento de trajectória francamente pobre. A figura 6.40 demonstra este efeito, a $1m/s$ e $1.5m/s$, respectivamente. Não foram executados testes a velocidades superiores porque o robot já se encontrava perigosamente perto de tombar para a frente. Nos casos em que a referência de orientação é fixa durante toda a trajectória o controlador preditivo leva então clara vantagem sobre o reactivo, ao prever o canto e desacelerar durante a sua execução, evitando travagens bruscas a realizando a curva "por dentro".



(a) $V_{ref} = 1m/s$ (b) $V_{ref} = 1.5m/s$

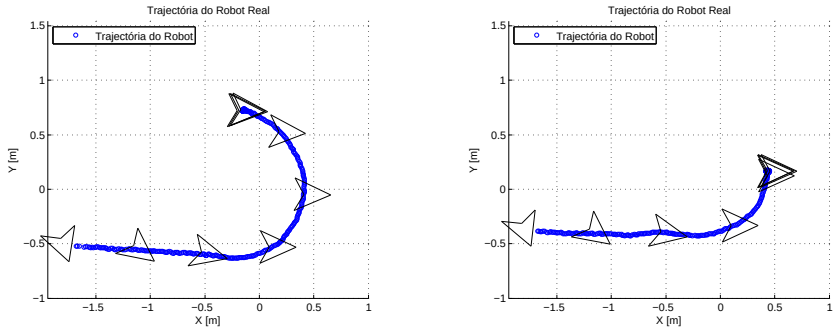
Figura 6.41: Ensaios com controlador reactivo, variação de θ_{ref} brusca

Nos casos em que a referência de orientação varia bruscamente no canto, o facto de o robot ter que rodar previne a ocorrência de uma travagem brusca e subsequente desequilíbrio descrito

anteriormente. Dado que todas as rodas se mantêm no chão a localização baseada em odometria mantém-se relativamente correcta e os resultados são menos catastróficos. Nestas situações, as trajectórias expostas na figura 6.41 foram observadas.

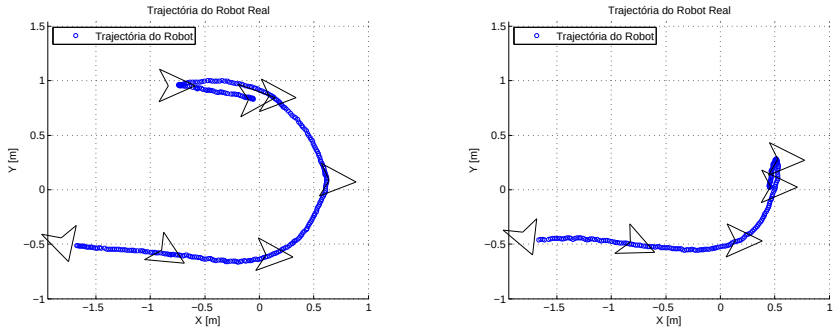
6.13.2.6 Controlador Preditivo, Trajectória em Gancho

Os mesmos ensaios que foram realizados em simulação com a trajectória em gancho foram repetidos com o robot real. Estes testes têm a particularidade de serem directamente comparáveis com os da simulação, dado que a trajectória de referência é idêntica. Os resultados obtidos apresentam-se de seguida, nas figuras 6.42, 6.43, e 6.44.



(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

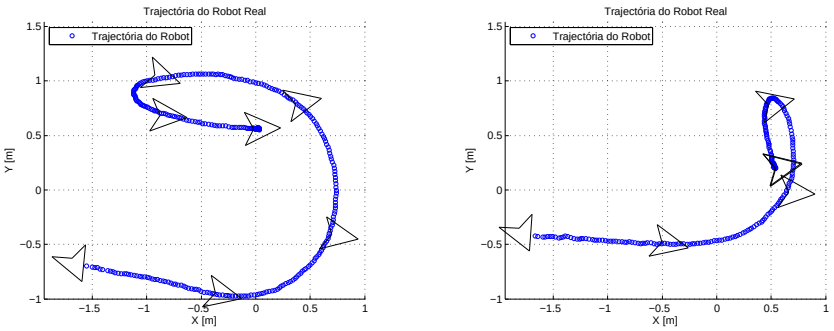
Figura 6.42: Trajectória em Gancho, $V_{ref} = 0.5m/s$



(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.43: Trajectória em Gancho, $V_{ref} = 0.75m/s$

Os resultados obtidos validam de certa forma o modelo utilizado para simular o robot. À semelhança do que foi observado em simulação, o robot é capaz de seguir a trajectória desejada a $0.5m/s$ sem problemas, degradando-se o seu desempenho progressivamente com o aumento da velocidade. A forma de geral das trajectórias observadas no robot real é idêntica à ocorrida em simulação, verificando-se que o robot real apresenta desvios algo maiores do ponto final antes de



(a) Gancho em semi-círculo (b) Gancho em quarto de círculo

Figura 6.44: Trajectória em Gancho, $V_{ref} = 1.0m/s$

o atingir. Isto pode ter várias explicações: por um lado, o modelo do robot em SimTwo, embora próximo, não é exactamente igual ao real. Por outro, o controlador está parametrizado para o robot simulado, parametrização essa que pode não ser idêntica para o robot real. Finalmente há ainda a considerar os erros de localização do robot real devido à odometria.

6.14 Conclusão

Neste capítulo foi descrita ao pormenor a estrutura e funcionamento do controlador preditivo para seguimento de trajectórias projectado no âmbito deste projecto. O algoritmo de controlo foi apresentado e explicitado, assim como todos os processos que ocorrem durante a execução deste (nomeadamente, o cálculo da trajectória de referência, a minimização da função custo, e a simulação da evolução do sistema). Foi também descrito o modelo simplificado utilizado para as predições feitas pelo controlador.

O controlador foi profusamente testado e o seu desempenho documentado e apresentado sob a forma de gráficos e tabelas. Numa primeira fase foram realizados ensaios com o intuito de encontrar os melhores parâmetros possíveis para o optimizador e modelo do sistema utilizados (secção 6.11). Porque a aproximação utilizada para o modelo do sistema é francamente simplificada, os seus parâmetros têm que ser adequados às velocidades pretendidas. Assim, encontrou-se uma relação matemática entre a constante de tempo ideal para o sistema de primeira ordem que modela o sistema e as velocidades de referência desejadas. De seguida realizaram-se uma grande quantidade de ensaios para determinar a influência do modelo do sistema utilizado e dos horizontes de previsão e controlo no desempenho do controlador. Os resultados foram apresentados e discutidos na secção 6.13. Destes ensaios surge que, dada a aplicação em causa e os limites de capacidade de processamento dos computadores utilizados, um horizonte de controlo de 2 ($N_u = 2$) e um horizonte de predição de 10 ($N_p = 10$) parecem fornecer a melhor relação desempenho/tempo de processamento possível. Os resultados apresentados documentam extensivamente o desempenho do controlador em condições que podem ser consideradas bastante exigentes.

Os ensaios realizados com o robot real utilizando ambos os controladores confirmam de um modo geral a superioridade do controlo preditivo. A capacidade de prever os cantos e assim desacelerar e virar mais cedo revelou-se um mais valia no seguimento de trajectórias, especialmente no caso em que a localização seja realizada puramente por odometria e haja pontos na trajectória que possam causar travagens bruscas. No entanto a precisão dos resultados obtidos (nomeadamente os erros de seguimento derivados do erro de localização do robot e o método de captura de resultados utilizado) não permitiu obter medidas de overshoot e tempo de estabelecimento directamente comparáveis com as obtidas em simulação.

Os resultados obtidos para o controlador revelaram-se de um modo geral bastante satisfatórios. Quando comparado com o controlador reactivo actualmente utilizado nos robots da equipa 5DPO, o controlador aqui projectado exibe um desempenho claramente superior, especialmente a velocidades elevadas. Esta superioridade vem à custa de um algoritmo bastante mais complexo e um tempo de processamento francamente mais elevado. Tentou-se no entanto manter o tempo de execução total do algoritmo abaixo dos $20ms$ de modo a não perturbar as tarefas de processamento de imagem. Empregando controlo preditivo, o robot revelou-se capaz de seguir com precisão elevada trajectórias com mudanças de direcção (cantos de em ângulo recto) e orientação a velocidades consideravelmente altas (até $2m/s$). Pela sua capacidade de ter em conta as limitações do sistema real (através da inclusão destas no modelo do sistema utilizado para as previsões) e de utilizar informação futura para prever os cantos da trajectória começando a virar antes de estes acontecerem, o controlo preditivo revelou-se uma excelente opção para situações em que desempenhos elevados a velocidades elevadas são necessários.

Capítulo 7

Controlador de Formação

Este capítulo detalha a estrutura e modo de funcionamento do controlador de formações desenvolvido, no contexto da sua utilização na equipa de futebol robótico. Um caso de estudo baseado numa situação afecta ao futebol robótico é apresentado, desenvolvido, e implementado, com o intuito de avaliar e ilustrar o desempenho do controlador.

7.1 Introdução

Tomando como base o controlador de trajectória desenvolvido iniciou-se o trabalho no controlo de formações. Definiu-se como objectivo para esta fase o projecto de uma *framework* adaptável baseada em controlo preditivo para criação e manutenção de formações de equipas de robots móveis, implementada no contexto do futebol robótico. As formações em causa não correspondem a formações absolutamente rígidas onde a posição relativa dos elementos da equipa deve ser mantida com a maior precisão, mas sim a formações ideais para optimizar a percepção conjunta, pela equipa, do meio ou de um elemento pertencente ao meio.

7.1.1 Definição e uso de Formações

Uma formação é usualmente definida como o arranjo espacial de um conjunto de agentes do mesmo tipo, onde as posições relativas dos seus elementos se mantêm fixas mesmo que a formação esteja em movimento. O movimento em formação traz habitualmente aos seus elementos algum tipo de benefício, usualmente de natureza táctica. O conceito de formação não é produto do engenho humano, dado que muito antes do homem sequer considerar problemas tácticos eram já observáveis comportamentos similares na natureza.

É habitual vislumbrar bandos de aves migratórias voando numa formação em V, o que lhes permite maximizar a distância viajada e minimizar o cansaço reduzindo o atrito do ar e voando no túnel de ar deixado pela ave que viaja imediatamente à frente (figura 7.1). Isto é verdade para todas as aves da formação excepto a que ocupa a posição da frente. No entanto, de modo a distribuir

a fadiga por entre os membros do bando, a ave que ocupa esta posição é trocada ciclicamente. Formações em coluna são também bastante comuns, por exemplo nos casos de uma ave que lidera uma coluna constituída pelas suas crias, o que permite à progenitora garantir um caminho seguro e às crias manter contacto visual com a ave da frente e evitar perder-se (muito habitual em patos).

Figura 7.1: Aves migratórias voando numa formação em V

Já na esfera humana as formações são utilizadas em situações diversas que vão desde operações militares a desportos de equipa. No contexto militar, habitualmente denominadas formações tácticas, são utilizadas no movimento coordenado de conjuntos de aviões, barcos, veículos terrestes, ou mesmo de soldados de infantaria. Nestes casos, a formação é utilizada com o intuito de obter uma vantagem táctica (i.e. formações em cunha de batalhões de infantaria são utilizadas para penetrar linhas de soldados inimigos ou por tropas especiais para carregar sobre massas de pessoas (7.2), formações em quadrado eram habitualmente utilizadas como protecção de cargas de cavalaria, formações em V com um elemento extra (a chamada *Finger-Four formation*) são habitualmente utilizadas por esquadrões de aviões em missões militares (7.2)).

Também nos desportos de equipa é feito uso extenso de formações. O futebol americano, por exemplo, é um desporto extremamente táctico onde as formações tomam um papel preponderante. Na realidade, existem mais de 30 que vão destes arranjos em T ou em I até arranjos extremamente complexos. No futebol, uma formação descreve o número de jogadores afectos a cada posição táctica no campo (genericamente, defesa, meio campo, e ataque). No caso deste desporto esta formação é menos rígida e não está em movimento, sendo mais uma atribuição de uma área de acção a cada jogador. Em desportos como volleyball ou andebol também são utilizadas formações para distribuição táctica dos jogadores.

Mais relacionado com o assunto desta tese, a utilização de formações em equipas de robots móveis é também bastante comum. No capítulo 2.3 foram referidas diversas estratégias para controlo de formações e as razões que as motivaram. Nos casos estudados anteriormente pretende-se manter um arranjo rígido de uma equipa de robots em que as posições relativas dos mesmos

(a) Formação Finger-four (b) Formação em Cunha

Figura 7.2: Formações tácticas militares

são fixas. No caso desta dissertação não é exactamente isso o que se pretende para as formações desenvolvidas. Pretende-se sim controlar formações que optimizem a percepção do meio por uma equipa de robots, sendo que para isso as suas posições relativas podem ter que variar no decurso do movimento da formação. O caso em estudo será descrito em pormenor posteriormente.

7.2 Tipos de Arquitecturas de Controlo

No que toca à estrutura mais geral do controlador preditivo, esta pode ser classificada em três tipos: *distribuída*, *centralizada*, ou *híbrida*. Estas categorias dizem respeito ao modo como os sinais de controlo de cada robot (e portanto, do comportamento da formação como um todo) são calculados.

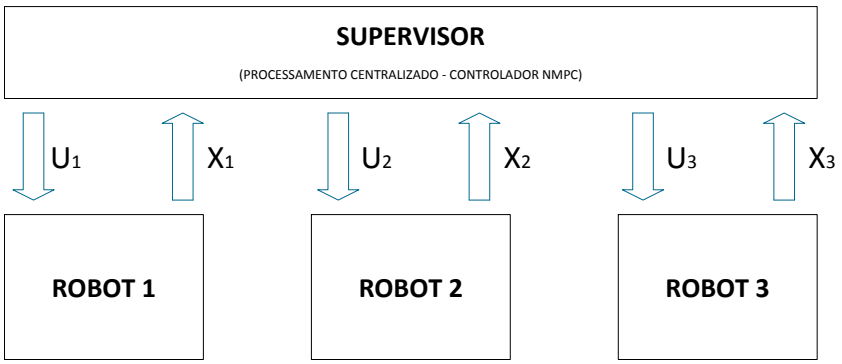


Figura 7.3: Esquema de um controlador MPC com arquitectura centralizada

Considere-se primeiro uma arquitectura completamente centralizada (figura 7.3), onde o algoritmo de controlo é executado numa máquina central e apenas os sinais de controlo U_n são enviados para cada robot. Cada robot por sua vez envia o seu estado X_n à unidade central. Esta

estratégia é a maneira mais simples de garantir a convergência da formação dado que, como todos os sinais de controlo são calculados ao mesmo tempo e localmente, o algoritmo prevê exactamente a evolução de cada elemento da equipa e calcula as entradas para cada um em concordância. Tem, no entanto, duas grandes desvantagens. A primeira prende-se com o elevado peso computacional de um controlador deste género dado que por cada robot que se adiciona à formação o número de entradas de controlo a optimizar cresce de forma proporcional ao horizonte de controlo N_c. Da necessidade de realizar uma simulação dinâmica para todos os robots surge ou a necessidade de muito poder de processamento ou a necessidade de diminuir os horizontes de predição e controlo. Para mais, uma arquitectura completamente centralizada é completamente intolerante a falhas: no caso de falha das comunicações ou da unidade de processamento central, toda a formação fica inutilizada.

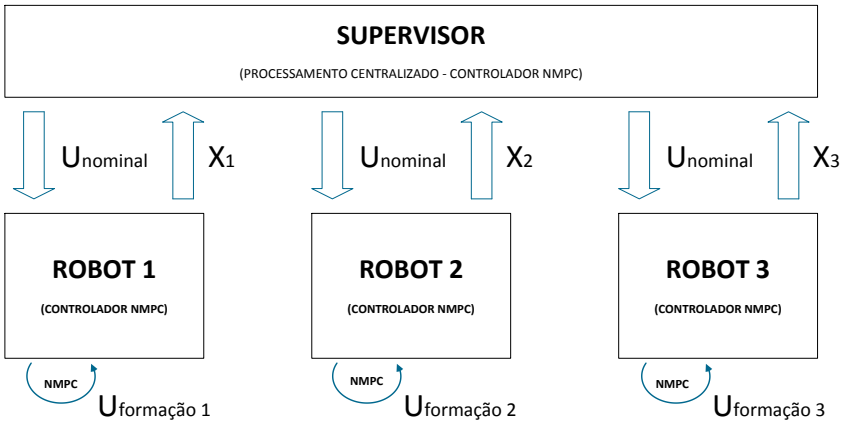


Figura 7.4: Esquema de um controlador MPC com arquitectura híbrida

Uma arquitectura híbrida (figura 7.4) toma partido do facto de que de um modo geral, uma vez criada a formação, os sinais de controlo nominais são iguais para todos os robots (à excepção, por exemplo, de um caso em que a equipa está a executar uma trajectória em arco e os arcos que cada robot descreve têm raios diferentes). Estas entradas de controlo nominais $U_{nominal}$ são calculadas centralmente e comunicadas a cada um dos robots. Estes, por sua vez, computam os sinais de controlo necessários para compensar os pequenos desvios em relação à trajectória de referência $U_{formacao}$ de forma a manter a formação. Esta arquitectura tem a grande vantagem de garantir até certo ponto a estabilidade da formação e ao mesmo tempo aliviar o peso de processamento na unidade central e em cada um dos robots. Sofre no entanto do mesmo mal das arquitecturas centralizadas: uma falha nas comunicações ou do sistema central é catastrófica.

Finalmente, há a considerar uma arquitectura plenamente distribuída (figura 7.5). Neste caso cada um dos robots calcula a totalidade das suas entradas de controlo U_n resolvendo o seu próprio problema de optimização. Isto retira a dependência de uma unidade de processamento central, garantindo o funcionamento da formação mesmo em caso de falha do sistema de comunicação. Cada robot necessita de ter, no entanto, informação sobre o estado X_n (posição e velocidade) dos

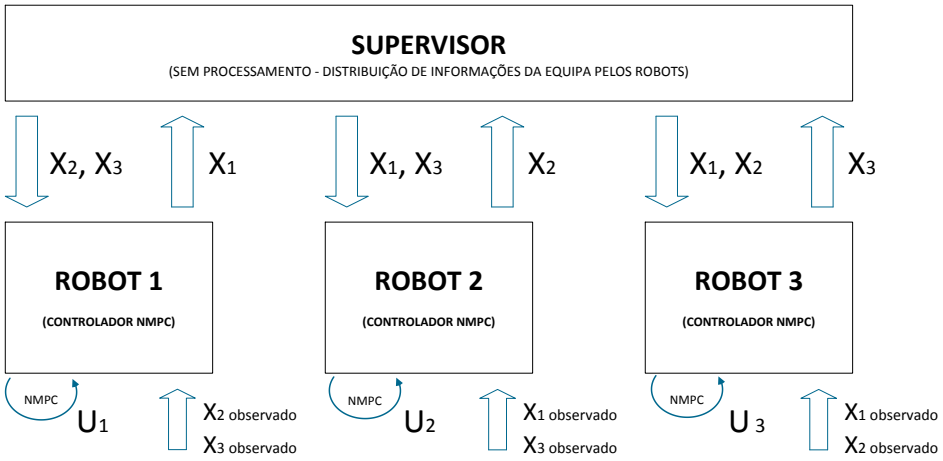


Figura 7.5: Esquema de um controlador MPC com arquitectura distribuída

companheiros de equipa. O controlador só será completamente distribuído se cada robot for capaz de obter esta informação por si próprio (e.g. através de um sistema baseado em visão ou recebendo esta informação directamente dos outros robots). Com esta arquitectura pode ser tomada uma aproximação mista, em que cada robot envia a sua própria posição e velocidade a um supervisor e recebe deste as informações sobre o estado dos colegas de equipa. Se os robots estiverem em permanente comunicação uns com os outros e tiverem capacidade de broadcast para tal o supervisor pode mesmo ser eliminado, e cada robot pode comunicar o seu estado directamente aos restantes elementos da equipa. No caso de falha das comunicações ou na máquina do supervisor, o robot utiliza os seus próprios meios para determinar estas informações, possuindo assim um certo grau de tolerância a falhas. Tem a desvantagem de colocar algum peso de processamento sobre cada um dos robots, porque cada um deles tem que simular a progressão da formação inteira. No entanto isto não se torna demasiado problemático porque cada robot só calcula as suas entradas de controlo. Esta arquitectura totalmente descentralizada torna também difícil garantir a estabilidade da formação, dado que cada robot resolve o seu problema de optimização isoladamente.

7.3 Estratégia de Controlo

As características do projecto ditaram a escolha de uma arquitectura plenamente descentralizada para o controlador de formações. O facto de existir um poder de processamento razoável em cada um dos robots utilizados (cada robot tem o seu próprio computador) permitiu a implementação deste tipo de arquitectura, dado que cada robot possui capacidade suficiente para determinar os seus próprios sinais de controlo utilizando NMPC. Para mais, toda a infra-estrutura de software (a existência de um programa supervisor em comunicação permanente com todos os robots da formação, facilmente adaptável para distribuir a posição de cada um dos robots pela formação) e hardware (os robots possuem um sistema de visão com o qual se auto-localizam, para além de

serem capazes de localizar a bola) se adequava perfeitamente à implementação de uma arquitectura como a descrita na figura 7.5.

7.3.1 Método de Controlo da Formação

A capacidade do controlador NMPC projectado de criar e manter uma formação advém do facto de as funções custo utilizadas pelos controladores de cada um dos robots da equipa estarem acopladas. Estão acopladas na medida em que as informações sobre a posição e velocidade dos restantes robots da formação são utilizadas na função custo de cada robot de forma a penalizar o desvio da geometria ou objectivo pretendidos, efectivamente tornando os robots um grupo coeso onde as acções de cada robot afectam os restantes companheiros.

Na arquitectura aqui projectada cada formação é então definida por um conjunto de funções custo mutuamente acopladas que penalizam essencialmente o desvio da posição e orientação do robot da sua posição desejada na formação. Note-se que os critérios de penalização não estão no entanto limitados à posição geométrica do robot, mas podem por exemplo impor que o vector velocidade de um robot seja paralelo ao vector velocidade de um dado alvo ou penalizar a proximidade dos outros robots de modo a evitar colisões. A cada um dos robots é atribuída a função custo associada à sua posição na formação.

7.4 Estrutura do Controlador

Estruturalmente, o controlador de formação é uma evolução e adaptação do controlador de trajectória apresentado no capítulo anterior. A figura 7.6 expõe a estrutura do controlador, sendo as semelhanças com o controlador de trajectória da figura 6.2 bastante claras. Para além do desaparecimento do elemento responsável por calcular a trajectória de referência para o controlador, vários elementos foram adicionados ou modificados:

- **Estado da Formação** - O controlador contém estruturas para guardar o estado da formação (posição e velocidade dos restantes robots da equipa ou de algum alvo que esteja a ser seguido), actualizadas a cada ciclo de controlo. Estas informações podem ser recebidas de um supervisor ou dos companheiros de equipa ou então determinadas pelo próprio robot utilizando o seus recursos.

- **Optimizador** - À semelhança do elemento correspondente do controlador de trajectória, utiliza um método de minimização numérica para optimizar a função custo e obter os sinais de controlo óptimos. Foi no entanto implementado um novo método denominado *Resilient Propagation* (RPROP), dado que se verificou a insuficiência do método do gradiente descendente para esta aplicação. Este método é explicado posteriormente.

- **Simulador** - O simulador foi expandido de modo a simular não só a evolução do seu próprio estado mas também do estado dos restantes elementos da formação (sejam eles robots ou alvos). Este elemento utiliza o modelo dinâmico simplificado desenvolvido no capítulo anterior para simular a sua própria evolução e modelos puramente cinemáticos para os restantes elementos. A velocidade dos restantes robots ou alvos é assumida durante todo o horizonte de predição como sendo constante constante e igual à velocidade actual.

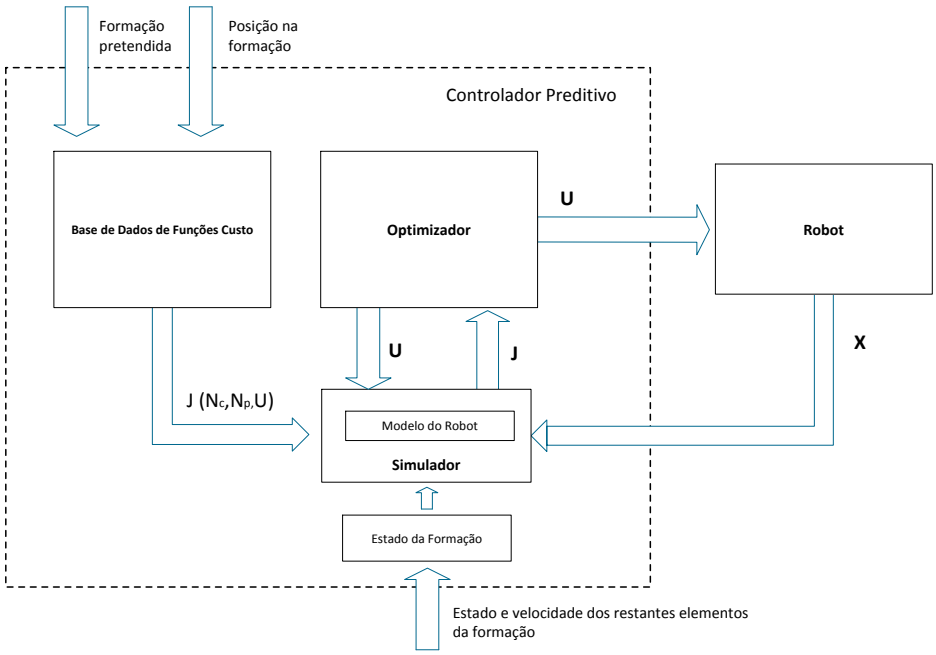


Figura 7.6: Estrutura do Controlador de Formação projectado

O controlador recebe como parâmetros a formação pretendida, para a qual foram definidas previamente as funções custo, e a posição do robot na formação. Destes escolhe a função custo correcta a utilizar (o que pode equivaler a uma função custo específica ou a uma parametrização de uma função mais geral). O controlador recebe também o estado actual dos restantes elementos da formação. O optimizador e o simulador trabalham depois em conjunto para determinar as entradas do sistema. O optimizador começa por fornecer ao simulador uma entrada de controlo U, e este por vez simula a evolução da formação completa para o horizonte de predição T_p. O simulador devolve ao optimizador um valor de custo para estas entradas de controlo, e o processo iterativo de minimização é repetido ciclicamente.

7.5 Algoritmo de Controlo

O algoritmo de controlo é em tudo semelhante ao utilizado no controlador de trajectória e descrito na secção 6.4, pelo que não será repetido aqui. As única alterações a apontar são que o cálculo da trajectória de referência foi removido e a simulação da evolução do estado do robot foi expandida para simular a formação inteira. O algoritmo de minimização foi também reprogramado, mas em termos de fluxo o algoritmo de controlo como um todo permanece igual.

7.6 Algoritmo de Minimização

Os testes iniciais foram realizados com o mesmo algoritmo de minimização empregue no controlador de trajectória, o método *Steepest Descent* descrito na secção 6.9. Com as funções custo utilizadas este apresentava alguns problemas de convergência quando os robots se encontravam muito distantes da posição onde deveriam estar, resultando num movimento lento ou simplesmente errático. Tomou-se esta oportunidade para implementar um algoritmo de optimização diferente, mais complexo mas mais eficiente que o *Steepest Descent*.

O algoritmo *Resilient Propagation* (RPROP) surgiu no contexto de algoritmos de aprendizagem para redes neuronais (ver [45]), tendo sido adaptado para utilização nesta aplicação. É um método adaptativo, onde o valor do passo não é proporcional ao valor do gradiente da função a minimizar em determinado ponto (como é o caso dos algoritmos de gradiente descendente) mas vai sendo adaptado conforme o comportamento da função. Consegue desta forma ser imune à imprevisibilidade do valor da derivada da função, dependendo apenas do comportamento temporal do seu sinal.

Seja $f(W)$ a função a minimizar, onde $W = [w_0, w_1, .., w_n]^T$ são as variáveis independentes. η^+ e η^- são parâmetros do optimizador e Δw_i o passo da variável $w_i(t)$ na iteração actual t. O algoritmo, conforme implementado, está definido de seguida:

Algorithm 3 Resilient Propagation (RPROP)

for i:= 0 to n **do**
  if $\frac{\partial f}{\partial w_i}(t-1) \cdot \frac{\partial f}{\partial w_i}(t) > 0$ **then**
    $\Delta_i(t) = min(\Delta_i(t-1) * \eta^+, \Delta_{max})$
    $\Delta w_i(t) = -sign(\frac{\partial f}{\partial w_i}(t)) * \Delta_i(t)$
    $w_i(t+1) = w_i(t) + \Delta w_i(t)$
  else if $\frac{\partial f}{\partial w_i}(t-1) \cdot \frac{\partial f}{\partial w_i}(t) < 0$ **then**
    $\Delta_i(t) = max(\Delta_i(t-1) * \eta^-, \Delta_{min})$
    $w_i(t+1) = w_i(t) - w_i(t-1)$
    $\frac{\partial f}{\partial w_i}(t) = 0$
  else if $\frac{\partial f}{\partial w_i}(t-1) \cdot \frac{\partial f}{\partial w_i}(t) = 0$ **then**
    $\Delta w_i(t) = -sign(\frac{\partial f}{\partial w_i}(t)) * \Delta_i(t)$
    $w_i(t+1) = w_i(t) + \Delta w_i(t)$
  end if
end for

Este algoritmo foi testado inicialmente com os valores sugeridos em [45] ($\eta^+ = 1.5$, $\eta^- = 0.5$, $\Delta_0 = 0.1$) e revelou ser capaz de convergir onde o *Steepest Descent* falhava. Estes resultados serão apresentados mais tarde.

7.7 Implementação da formação

Este sub-capítulo descreve a implementação de uma formação de três robots em torno de um alvo utilizando a framework definida atrás. É apresentada a definição do problema e os objectivos a atingir com a formação, sendo estudada a formação ideal para os cumprir e as funções custo que a implementam. São ainda apresentados resultados de ensaios em simulação, com o intuito de testar e documentar o desempenho do controlador.

7.7.1 Objectivo da Formação

Deseja-se implementar uma formação constituída por três robots da equipa de futebol robótico 5DPO que cumpra dois objectivos: optimizar a percepção da velocidade da bola utilizando dois robots enquanto que um terceiro se posiciona no local ideal para realizar a recepção da mesma. Os robots devem manter esta formação acompanhado o movimento da bola e evitando ao máximo colisões com ela ou com os colegas de equipa.

7.7.2 Definição matemática da Formação

A definição matemática do sistema obtida na secção 6.2 pode agora ser extendida à formação completa, contendo uma bola e três robots. Tome-se como base para definição da formação os elementos representados na figura 7.7.

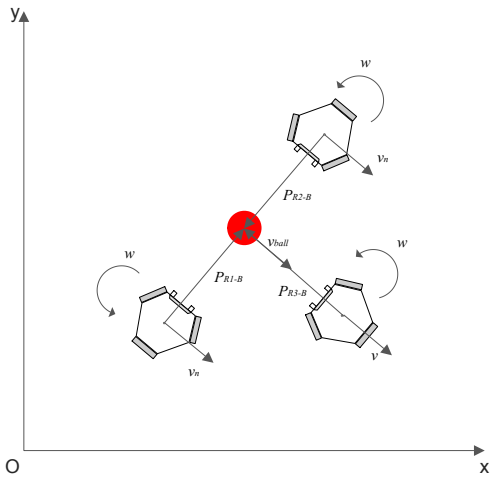


Figura 7.7: Formação completa, com três robots e uma bola

Sejam então X_{ball} e v_{ball} os vectores posição e velocidade da bola, respectivamente, em coordenadas globais:

$$X_{ball}(k) = [\, x_{ball}(k) \quad y_{ball}(k) \,]^T \tag{7.1}$$
$$v_{ball}(k) = [\, vx_{ball}(k) \quad vy_{ball}(k) \,]^T \tag{7.2}$$

Considere-se também o versor que representa o sentido e direcção da velocidade da bola $\hat{v}_{ball}(k) = [\, \hat{vx}_{ball}(k) \quad \hat{vy}_{ball}(k) \,]^T$, tal que:

$$\hat{v}_{ball}(k) = \frac{v_{ball}(k)}{\sqrt{vx^2_{ball}(k) + vy^2_{ball}(k)}} \tag{7.3}$$

Para cada robot n tem-se que o seu estado é representado por:

$$X_n(k) = [\, x_n(k) \quad y_n(k) \quad \theta_n(k) \,]^T \tag{7.4}$$
$$V_n(k) = [\, vx_n(k) \quad vy_n(k) \quad w_n(k) \,]^T \tag{7.5}$$

A posição da bola em relação a cada robot é dada pelo vector $P_{Rn-B}(k) = [\, x_{Rn-B}(k) \quad y_{Rn-B}(k) \,]$, onde:

$$P_{Rn-B}(k) = [\, (x_{ball}(k) - x_n(k)) \quad (y_{ball}(k) - y_n(k)) \,] \tag{7.6}$$

Defina-se também o versor $\hat{P}_{Rn-B}(k) = [\, \hat{x}_{Rn-B}(k)\ \hat{y}_{Rn-B}(k) \,]$, que indica a direcção da bola em relação ao robot, e o seu ângulo θ_{Rb-B}:

$$\hat{P}_{Rn-B}(k) = \frac{P_{Rn-B}(k)}{\sqrt{x^2_{Rn-B}(k) + y^2_{Rn-B}(k)}} \tag{7.7}$$
$$\theta_{Rb-B}(k) = atan2(\, x^2_{Rn-B}(k)\ ,\ y^2_{Rn-B}(k)\,) \tag{7.8}$$

Finalmente, estão também definidas as posições de cada robot n em relação aos companheiros de formação y, dadas por $P_{Rn-Ry}(k) = [\, x_{Rn-Ry}(k)\ y_{Rn-Ry}(k) \,]$ onde:

$$P_{Rn-Ry}(k) = [\, (x_n(k) - x_y(k)) \quad (y_n(k) - y_y(k)) \,] \tag{7.9}$$

7.7.3 Formação Ideal

De seguida é descrita a formação ideal de modo a cumprir os objectivos delineados anteriormente. Os dois objectivos principais são estudados separadamente.

7.7.3.1 Percepção da velocidade da Bola

A qualidade da estimação do estado da bola é uma função da sua direcção de movimento em relação ao robot e da sua distância ao mesmo. Esta estimação é realizada utilizando um sistema de visão omnidireccional cuja descrição pode ser encontrada em [35]. Uma vez que o sistema de visão é omnidireccional a orientação do robot é irrelevante neste caso.

A distância do robot à bola influencia o número de píxeis que o robot vê desta. Distâncias demasiado grandes levam a que o robot não detecte a bola correctamente e falhe assim a estimativa da sua posição e velocidade. Quando a distância é demasiado pequena pode ocorrer que o robot não veja a bola completa e assim seja também incapaz de detectar correctamente a sua posição, para além de existir o risco de uma colisão indesejada. De modo a garantir a melhor estimativa possível da posição e velocidade da bola o robot deve manter desta uma distância fixa, nem muito grande nem muito pequena.

A posição do robot em relação à bola, encontrando-se esta em movimento, também influencia a qualidade da percepção da sua velocidade. Tome-se por exemplo o caso em que a bola se aproxima do robot em linha recta, na direcção exacta deste. Neste caso o robot apenas vê o tamanho da bola a aumentar, tornando-se difícil a estimação da velocidade. O caso ideal corresponde àquele em que a bola se movimenta numa direcção perpendicular à sua posição em relação ao robot.

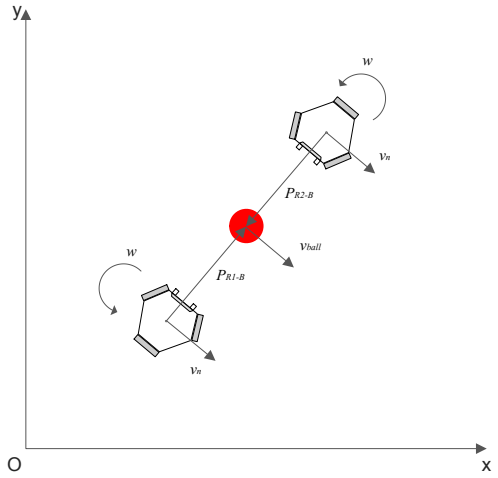


Figura 7.8: Formação ideal de dois robots em torno da bola para estimar a sua velocidade

Assim, a formação ideal de dois robots em torno da bola de modo a melhor estimar a velocidade da mesma é apresentada na figura 7.8 e possui as seguintes características:

- Cada um dos robots posiciona-se de um dos lados da bola, mantendo uma velocidade paralela à velocidade da mesma, v_{ball}, e com o mesmo sentido e módulo desta.
- O vector posição do robot em relação à bola P_{Rx_B} deve ser perpendicular ao vector velocidade da bola v_{ball}.

- Cada um dos robots deve manter uma distância fixa $|P_{Rx_B}|$ da bola.
- Os robots não devem colidir entre si.

7.7.3.2 Recepção da bola

A posição ideal do robot em relação à bola para preparar uma boa recepção (em que a bola é correctamente dominada, não ressaltando no kicker) corresponde àquela em que o vector velocidade do robot é colinear com o vector velocidade da bola, com o mesmo módulo e sentido, e a orientação do robot é tal que a parte da frente esteja virada para a bola. O robot pode então ser suavemente desacelerado e a distância à bola encurtada de modo a efectuar a recepção nas condições ideais. Esta formação é apresentada na figura 7.9:

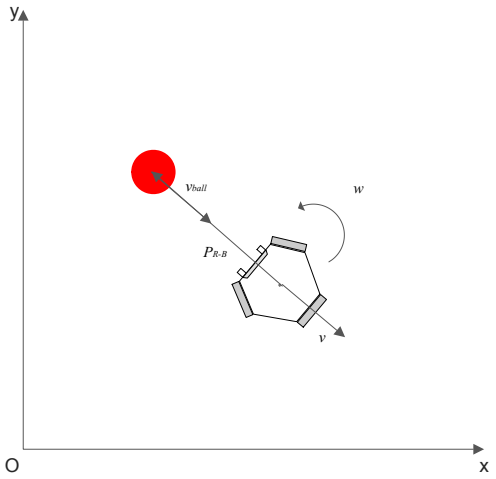


Figura 7.9: Formação ideal de um robot e bola de modo a optimizar a recepção da mesma

Sistematizando, a formação descrita possui as seguintes características:

- A velocidade do robot deve ser igual em módulo, direcção, e sentido à velocidade da bola v_{ball}.
- O vector posição do robot em relação à bola P_{Rn-B} deve ser colinear com o vector velocidade da bola v_{ball}.
- A orientação do robot θ_n deve a todo o momento ser igual ao ângulo do vector P_{Rn-B}, definido por θ_{Rn-B}, de modo a que o kicker do robot esteja sempre virado para a bola.
- O robot deve manter uma distância fixa $|P_{R-B}|$ da bola.

7.7.3.3 Formação completa

A formação completa é constituída pelos três robots e a bola. Devem ser garantidas todas as características descritas anteriormente para as tarefas de estimação da velocidade e recepção

da bola, evitando ainda que os três robots colidam entre si ou com a bola. A figura 7.7 ilustra a formação completa.

7.7.4 Funções Custo

Uma vez delineada a formação pretendida e obtida uma representação matemática da formação devem ser definidas as funções custo necessárias para que cada robot tenda para a posição que lhe compete. Estas devem penalizar o desvio do estado de cada robot das situações ideais referidas em 7.7.3.1 e 7.7.3.2.

Cada função custo é constituída por vários elementos, cada um essencialmente uma função quadrática de variáveis representativas do estado da formação. Cada um destes elementos penaliza o desvio do robot da sua posição ideal na formação.

7.7.4.1 Posição 1 - Percepção da Velocidade da Bola

Esta é a função custo atribuída aos dois robots que se vão colocar dos lados da bola, de maneira a obter a melhor estimativa possível da sua velocidade. Defina-se para cada uma das características pretendidas uma função matemática que penalize o desvio desta.

Distância do Robot em relação à Bola - A manutenção da distância pretendida $d_{set\,point}$ do robot n à bola pode ser garantida pelo termo quadrático:

$$f_1(N_1,N_2) = \sum_{i=N_1}^{N_2} (d_{set\,point} - |P_{Rn-B}(i)|)^2 \tag{7.10}$$

$|P_{Rn-B}(i)|$, módulo do vector posição do robot n em relação à bola, corresponde obviamente à distância a que este está da bola. A função simplesmente penaliza desvios desse valor de $d_{set\,point}$.

Velocidade e posição do Robot em relação à Bola - O desvio das características pretendidas, descritas em 7.7.3.1, pode ser penalizado de maneira muito simples por:

$$f_2(N_1,N_2) = \sum_{i=N_1}^{N_2} (\hat{P}_{Rn-B}(i) \cdot \hat{v}_{ball}(i))^2 \tag{7.11}$$

Garantir que os vectores da velocidade da bola v_{ball} e da posição do robot em relação à bola P_{Rn-B} sejam a todo o momento perpendiculares obriga ao mesmo tempo a que o robot mantenha a posição desejada em relação à bola e a que a sua velocidade seja igual à dela. É um resultado conhecido que o produto interno de dois vectores é nulo no caso de os vectores serem perpendiculares. O termo acima penaliza qualquer valor de produto interno dos dois vectores que seja maior que 0, efectivamente fazendo com que estes tendam a ser perpendiculares. Em vez de se utilizar directamente os vectores foram utilizados os versores correspondentes $\hat{v}_ball$ e $\hat{P}_{Rn-B}$ de modo a garantir que apenas a sua direcção tinha peso, e não o seu módulo.

Prevenção de Colisões - A colisão entre robots é evitada penalizando largamente a proximidade extrema entre estes. Sejam y_1 e y_2 o número dos colegas de equipa do robot n, e d_{min} a distância que um robot mede do outro quando se estão a tocar. Esta não é nula porque as distâncias são medidas entre centros geométricos e há que considerar a estrutura do robot. Temos então:

$$f_3(N_1,N_2) = \sum_{i=N_1}^{N_2} ((\frac{1}{-d_{min}+|P_{Rn-Ry_1}(i)|})^2 + (\frac{1}{-d_{min}+|P_{Rn-Ry_2}(i)|})^2) \tag{7.12}$$

A adição do termo d_{min} ao numerador da função desloca toda o gráfico para a direita, resultando em que f_3 tenda para infinito quando a distância entre robots se aproxima do valor equivalente a uma colisão. Isto corresponde a a uma penalização extrema de colisões entre robots.

Função custo completa - Tendo em conta todos os elementos atrás definidos, os pesos dados a cada um deles, e ainda um termo que penaliza a variação do esfoço de controlo, a função custo completa para os dois robots que devem calcular a velocidade da bola encontra-se definida como:

$$\begin{aligned} J(N_1,N_2,N_c) \; = \; & \sum_{i=N_1}^{N_2} \lambda_1 (d_{setpoint} - |P_{Rn-B}(i)|)^2 + \\ & \sum_{i=N_1}^{N_2} \lambda_2 (\hat{P}_{Rn-B}(i) \cdot \hat{v}_{ball}(i))^2 + \\ & \sum_{i=N_1}^{N_2} \lambda_3 ((\frac{1}{-d_{min}+|P_{Rn-Ry_1}(i)|})^2 + (\frac{1}{-d_{min}+|P_{Rn-Ry_2}(i)|})^2) + \\ & \sum_{i=1}^{N_c} \lambda_4 (\Delta U(i)) \end{aligned} \tag{7.13}$$

Onde:

- N_1 e N_2 são os limites do horizonte de predição que se pretende incluir no cálculo da função custo tal que $N_1 > 0$ e $N_2 \leq N_p$.
- N_c é o horizonte de controlo.
- λ_1, λ_2, λ_3, e λ_4 são os pesos associados a cada um dos componentes da função custo.
- $\Delta U(k) = [v(k)_{ref} - v_{ref}(k-1)]^2 + [vn_{ref}(k) - vn_{ref}(k-1)]^2 + [w_{ref}(k) - w_{ref}(k-1)]^2$, correspondendo à variação dos valores de controlo, sendo $U(i) = [\; v_{ref}(k)\; vn_{ref}(k)\; w_{ref}(k)\;]^T$ o vector das velocidades de referência.

7.7.4.2 Posição 2 - Recepção da Bola

De seguida define-se a função custo empregue pelo robot que se deve posicionar para receber a bola em condições ideais.

Distância do Robot em relação à Bola - Para melhor receber a bola, o robot receptor deve acompanhá-la a uma distância fixa que é progressivamente diminuída. À semelhança da função definida atrás, a manutenção da distância pretendida $d_{setpoint}$ do robot n à bola pode ser garantida pelo termo quadrático:

$$f_1(N_1,N_2) = \sum_{i=N_1}^{N_2} (d_{setpoint} - |P_{Rn-B}(i)|)^2 \tag{7.14}$$

Velocidade e posição do Robot em relação à Bola - O desvio da posição e velocidade ideais do robot definidas previamente pode ser penalizado pela função:

$$f_2(N_1,N_2) = \sum_{i=N_1}^{N_2} (\hat{P}_{Rn-B}(i) \cdot \hat{v}_{ball}(i) - (-1))^2 \tag{7.15}$$

Forçar o vector P_{Rn-B} a ser colinear e de direcção oposta ao vector v_{ball} pode ser conseguido garantindo que o produto interno dos versores $\hat{P}_{Rn-B}$ e v_{ball} seja igual a -1. A função quadrática acima definida penaliza desvios deste valor. A conjugação de f_2 com f_1 vai também conduzir a que a velocidade do robot seja igual em módulo, sentido, e direcção à velocidade da bola. Mal a bola encoste ao robot, o que é conseguido por diminuição progressiva de $d_{setpoint}$, o controlador passa a ser outro (não considerado neste momento).

Orientação do Robot - O robot deve ser mantido a todo o momento com a sua parte frontal virada para a bola, o que pode ser conseguido igualando a sua orientação ao ângulo do vector P_{Rn-B}. Esse critério está definido pela função seguinte:

$$f_3(N_1,N_2) = \sum_{i=N_1}^{N_2} (diffAngle(\theta_n, \theta_{Rn-B}))^2 \tag{7.16}$$

A função $diffAngle(x,y)$ retorna a diferença, em radianos, entre os ângulos x e y tal que $diffAngle(x,y) \in [-\pi,\pi]$. f_3 irá portanto penalizar qualquer diferença entre o ângulo do robot e o ângulo do vector da sua posição em relação à bola. Minimizar este erro resulta no robot tender a manter a sua parte frontal virada para a bola.

Função custo completa - Agregando todas as penalizações definidas anteriormente, multiplicando cada uma por um peso, e adicionando um termo que penalize a variação do esforço de

controlo, temos para o robot receptor da bola a seguinte função custo:

$$
\begin{aligned}
J(N_1,N_2,N_c) \;=\; & \sum_{i=N_1}^{N_2} \lambda_1 (d_{set\,point} - |P_{Rn-B}(i)|)^2 + \\
& \sum_{i=N_1}^{N_2} \lambda_2 (-1 + \hat{P}_{Rn-B}(i) \cdot \hat{v}_{ball}(i))^2 + \\
& \sum_{i=N_1}^{N_2} \lambda_3 ((\frac{1}{-d_{min} + |P_{Rn-Ry_1}(i)|})^2 + (\frac{1}{-d_{min} + |P_{Rn-Ry_2}(i)|})^2) + \\
& \sum_{i=N_1}^{N_2} \lambda_4 (diffAngle(\theta_n, \theta_{Rn-B}))^2 + \\
& \sum_{i=1}^{N_c} \lambda_5 (\Delta U(i))
\end{aligned}
\tag{7.17}
$$

Onde:

- N_1 e N_2 são os limites do horizonte de predição que se pretende incluir no cálculo da função custo tal que $N_1 > 0$ e $N_2 \leq N_p$.
- N_c é o horizonte de controlo.
- λ_1, λ_2, λ_3, λ_4, e λ_5 são os pesos associados a cada um dos componentes da função custo.
- $\Delta U(k) = [v(k)_{ref} - v_{ref}(k-1)]^2 + [vn_{ref}(k) - vn_{ref}(k-1)]^2 + [w_{ref}(k) - w_{ref}(k-1)]^2$, correspondendo à variação dos valores de controlo, sendo $U(i) = [\ v_{ref}(k)\ vn_{ref}(k)\ w_{ref}(k)\ ]^T$ o vector das velocidades de referência.

7.8 Preparação dos Ensaios do Controlador

Uma vez implementada a formação sobre a *framework* projectada procedeu-se a testes em simulação de modo a aferir o desempenho do controlador sob diferentes condições. Dois grandes blocos de testes foram realizados. O primeiro consiste em determinar a capacidade do controlador de obrigar os robots a convergir para a formação em torno da bola, partindo estes de posições distantes das que devem ocupar, a partir de diversos arranjos espaciais iniciais. Neste caso, a bola permanece estática. O segundo consiste em aferir a capacidade do controlador de manter a formação em torno da bola estando esta em movimento. Aqui são realizados ensaios forçando o movimento da bola segundo diversas trajectórias (que incluem rectas, arcos, e cantos), a diferentes velocidades.

7.8.1 Ambiente de Simulação

À semelhança dos resultados apresentados no capítulo anterior, o *SimTwo* foi mais uma vez utilizado para simular a formação, tendo-se criado um cenário com três robots omnidireccionais

idênticos ao anterior e uma bola. A complexidade extra de controlar três robots ao mesmo tempo que se corria a simulação também com três robots e o software supervisor para distribuir as informações da formação por cada cada robot obrigou ao uso de meios computacionais adicionais.

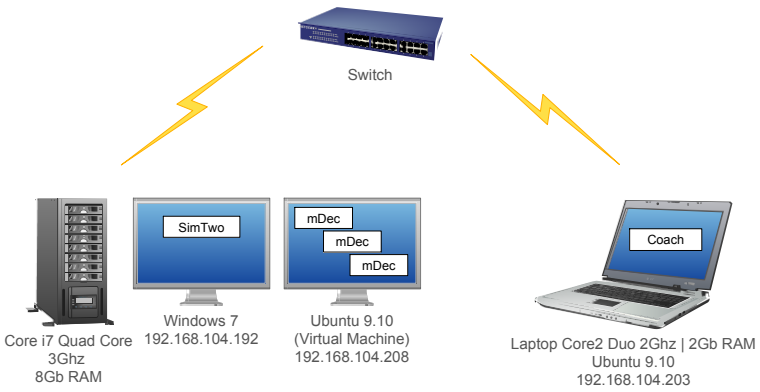


Figura 7.10: Setup de teste em simulação para o controlador de formações

Assim, foi montado um setup de testes com dois computadores em rede: uma workstation (Intel Core i7 @2.8Ghz/Core com 8Gb RAM) a correr Windows 7 nativamente e Ubuntu 9.10 em virtual machine (os dois sistemas operativos funcionam efectivamente como duas máquinas diferentes, tendo incluive IPs diferentes dentro da rede) e um computador portátil (Intel Core2 Duo @2Ghz/Core com 2Gb de RAM) também a correr Ubuntu 9.10. Este arranjo pode ser visto na figura 7.10. O SimTwo corre sobre o Windows 7 da workstation, enquanto que as três instâncias do controlador (uma para cada robot) correm em Ubuntu na mesma máquina, em processos separados. O portátil é utilizado para executar o Coach, a aplicação supervisora que é utilizada para distribuir por cada controlador o estado dos restantes robots da formação. As comunicações entre aplicações processam-se por UDP conforme ilustrado no diagrama da figura 7.11.

O SimTwo toma o papel do software de visão e interface com hardware (OVIS), enviando para cada mDec o estado do robot que lhe corresponde e da bola. Cada mDec, por sua vez, envia para o SimTwo as referências de controlo do seu robot. Cada mDec comunica também com o Coach, enviando o seu próprio estado e o estado da bola conforme a observa. Finalmente, o Coach envia a cada mDec individualmente o estado dos restantes robots da formação, de modo a que cada robot tenha informação da posição e velocidade dos seus companheiros de equipa. É de notar que este arranjo é em tudo semelhante ao utilizado nos robots reais, tomando o SimTwo a vez do OVIS (que processa a visão e comunica com os drivers dos motores, servindo de ponte entre o software e o hardware). Trocar o controlo do simulador para os robots reais é simplesmente uma questão de trocar, em cada mDec, o IP do SimTwo pelo IP da máquina que está a correr o OVIS no robot real correspondente.

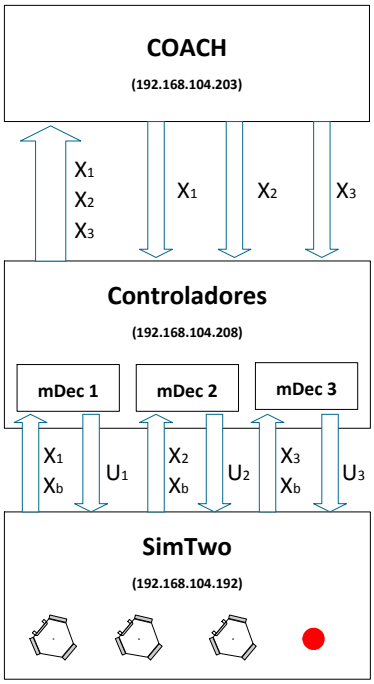


Figura 7.11: Diagrama das comunicações entre aplicações

7.8.2 Parâmetros de Teste

Entre os pesos de cada elemento presente na função custo e os parâmetros do optimizador há diversas variáveis que influenciam a qualidade dos resultados. De seguida apresentam-se os valores utilizados para cada uma delas.

Variável	Valor
λ_1	10
λ_2	100
λ_3	100
λ_4	5

Tabela 7.8.1: Pesos da função custo dos robots estimadores da velocidade

Variável	Valor
λ_1	10
λ_2	100
λ_3	1000
λ_4	50
λ_5	5

Tabela 7.8.2: Pesos da função custo do robot receptor da bola

No caso da minimização das funções custo apenas a relação entre os pesos dados aos diversos elementos importa para o resultado final e não o seu valor absoluto, pelo que por uma questão de simplicidade estes foram mantidos inteiros. Assim, deu-se à penalização da posição relativa em relação à bola λ_2 um valor dez vezes superior à penalização da distância à mesma λ_1, dado que a primeira é mais complicada de manter. λ_3, o peso dado à penalização da proximidade entre robots, foi mantido elevado de modo a evitar ao máximo a ocorrência de colisões. A penalização

da orientação do robot receptor, λ_4 na tabela 7.8.2, foi mantida a metade do valor da penalização de posição, por não ser absolutamente essencial (o robot pode receber a bola em qualquer orientação e rodar de seguida para a rematar). O valor de penalização da variação do esforço de controlo (λ_4 na primeira tabela e λ_5 na segunda) foi escolhido como sendo o valor mais pequeno que evitava que os robots se movessem em torno da bola quando esta estivesse estacionária. Os valores finais a que se chegou foram resultado de um processo iterativo, em muito facilitado devido à simplicidade com que os pesos podem ser alterados a partir da interface implementada. Este processo não necessita de ser particularmente preciso, dado que uma gama alargada de pesos fornece resultados semelhantes.

Variável	Valor
IT_{max}	20
ε	0.05
η^+	1.2
η^-	0.8

Tabela 7.8.3: Parâmetros do optimizador RPROP

Os parâmetros inicialmente utilizados para o RPROP correspondem aos sugeridos no artigo de onde se retirou a descrição do algoritmo [45]. Os primeiros testes realizados com estes parâmetros ($\eta^+ = 1.5$ e $\eta^- = 0.5$) resultaram de imediato em desempenhos bastante satisfatórios. Algumas alterações testadas em torno destes valores (decremento de η^+ e incremento de η^-) resultaram em melhorias directamente observáveis, tendo-se por fim chegado aos valores descritos na tabela 7.8.2. Existe bastante flexibilidade e robustez no ajuste dos parâmetros do controlador de um modo geral, o que é muito positivo.

7.8.3 Formato dos Resultados

Os resultados serão apresentados sob a forma de plots X/Y das trajectórias dos três robots e da bola. A intervalos de tempo definidas são marcados sobre a trajectória ícones representativos do robot e da bola, de modo a fornecer uma ideia da geometria da formação a cada momento. A bola é representada por uma bola vermelha e cada um dos robots por um símbolo que pode ser visto na figura 7.12.

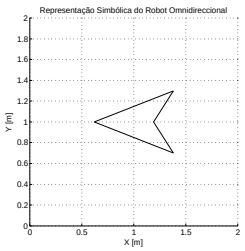


Figura 7.12: Símbolo utilizado para representar um robot omni-direccional

Cada robot é representado por um símbolo idêntico ao da figura 7.12. A parte côncava do triângulo corresponde à frente do robot, que tem o kicker e os apoios para recepção da bola. Nos resultados apresentados, sobre cada símbolo da bola e dos robots são também marcados os vectores velocidade de cada elemento nesse instante de tempo.

7.9 Ensaios do Controlador

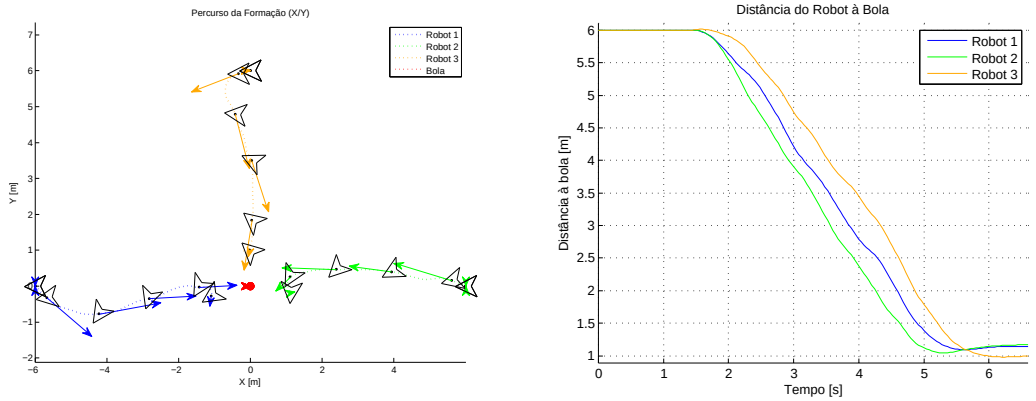
De seguida reproduzem-se os resultados obtidos para os ensaios em simulação feitos com o controlador de trajectória. São primeiro apresentados os ensaios para teste da convergência da equipa para a formação pretendida, partindo de configurações geométricas iniciais diversas.

7.9.1 Convergência para Formação

Os resultados seguintes demonstram as trajectórias percorridas por cada um dos robots quando, partindo de posições diferentes, se dá a ordem para criar a formação. A bola permanece estacionária durante cada ensaio, pelo que o produto interno de qualquer vector com o vector velocidade da bola é nulo. Os robots 1 e 2 correspondem aos agentes responsáveis por estimar a velocidade da bola enquanto que o robot 3 é o receptor. As distâncias pretendidas de cada robot à bola foram definidas para 1 m.

7.9.1.1 Ensaio 1

Neste ensaio os robots partem de posições perfeitamente opostas da bola. Por haver menor risco de colisões ou possibilidade dos robots se atrapalharem mutuamente, é o caso mais simples. Os resultados podem ser vistos na figura 7.13.



(a) Plot XY

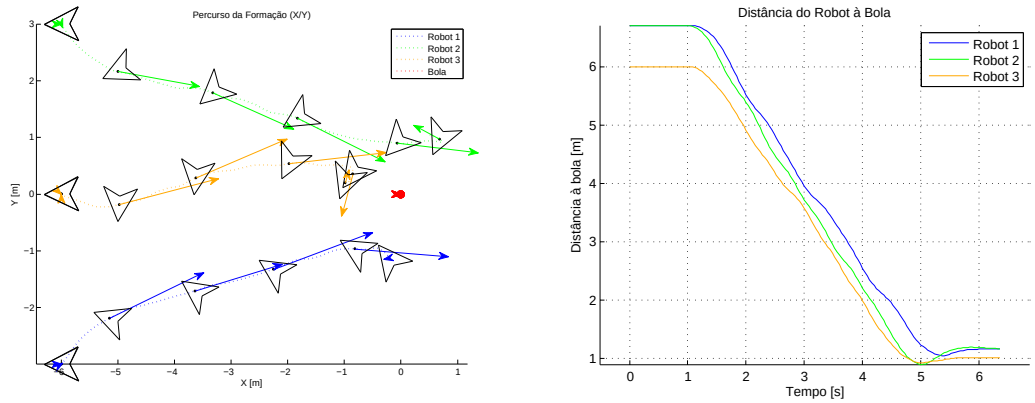
(b) Distância à bola

Figura 7.13: Convergência para formação, ensaio 1

Os robots tendem perfeitamente para as suas posições na formação, realizando trajectórias quase rectas até esta. Observando o plot da distância em função do tempo estima-se que os robots tenham convergido para a formação desejada em cerca de quatro segundos.

7.9.1.2 Ensaio 2

Aqui todos os robots partem do mesmo lado da bola, embora separados por uma distância de 3 metros. Os resultados podem ser vistos na figura 7.14.



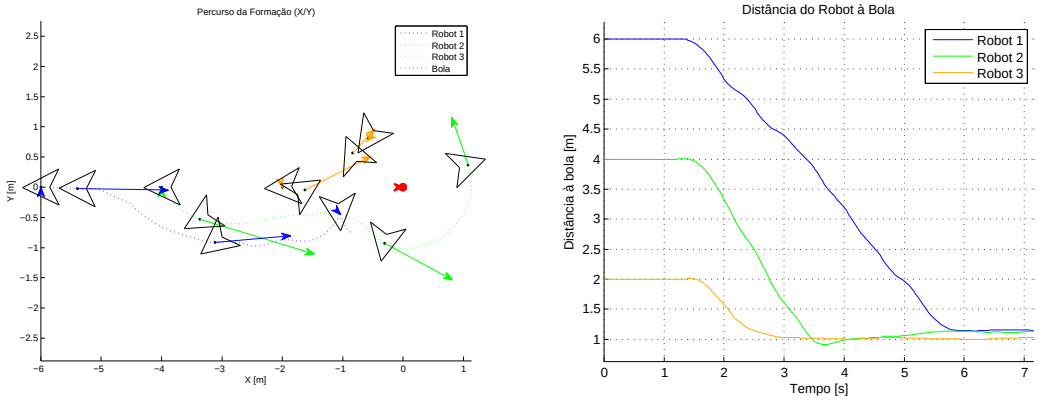
(a) Plot XY (b) Distância à bola

Figura 7.14: Convergência para formação, ensaio 2

Aqui começa a ser observável alguma interacção entre robots. O robot 3 progrediu imediatamente para a bola, em linha recta. Os robots 1 e 2 seguiram também em direcção à bola afastando-se do robot 3 quando se começaram a aproximar, acabando por se colocar um de cada lado desta. A formação convergiu em cerca de 5 segundos.

7.9.1.3 Ensaio 3

O terceiro ensaio ilustra a situação mais complicada, em que os robots partem todos no mesmo alinhamento em relação à bola. Os resultados apresentam-se na figura 7.15.



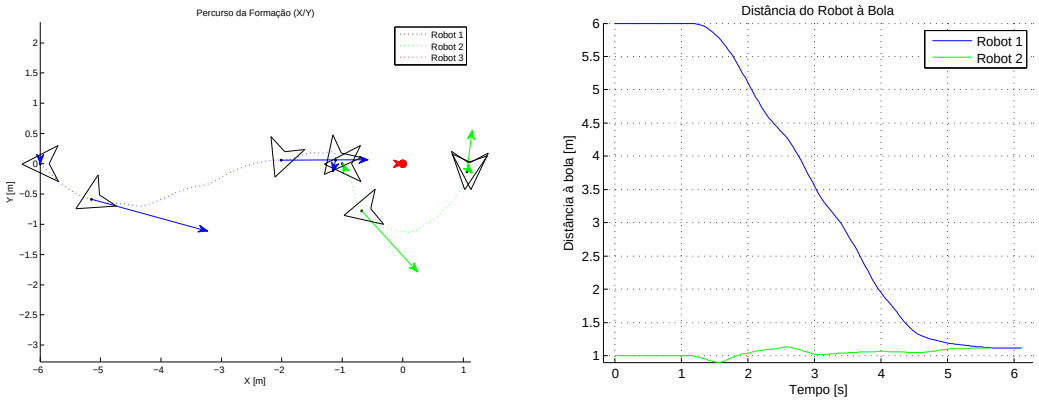
(a) Plot XY (b) Distância à bola

Figura 7.15: Convergência para formação, ensaio 3

O robot 3, estando mais perto da bola, tende a ir ocupar a posição imediatamente à sua frente. Com a aproximação do robot 2 desloca-se ligeiramente para cima, rodando em torno da bola, indo este ocupar o espaço imediatamente atrás dela. Finalmente, o robot 1 posiciona-se no espaço deixado entre os robots 2 e 3. Este processo demora cerca de 4.5 segundos.

7.9.1.4 Ensaio 4

Este ensaio utiliza apenas os robots 1 e 2, e ilustra o que acontece quando um robot já está na sua posição desejada em relação à bola e é dada ordem a outro robot para se juntar à formação. Os resultados podem ser vistos na figura 7.16.



(a) Plot XY

(b) Distância à bola

Figura 7.16: Convergência para formação, ensaio 4

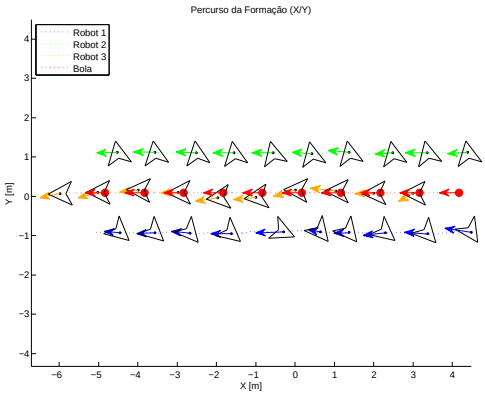
Neste caso o robot 2 encontra-se já na posição desejada em relação à bola. Com a aproximação do robot 1 roda em torno da bola indo ocupar a posição imediatamente oposta, enquanto que o robot 1 toma a sua posição anterior.

7.9.2 Manutenção da Formação

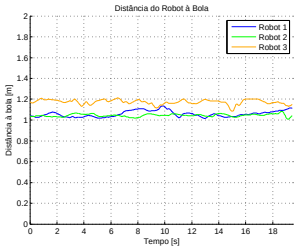
Os ensaios seguintes testam situações em que, uma vez formada a geometria desejada para a formação, se movimenta a bola segundo diversas trajectórias de modo a avaliar a capacidade do controlador de manter as posições desejadas. Os vários ensaios correspondem aos vários tipos de trajectória testados. São apresentados plots XY, gráficos de distância à bola e de produto interno entre os versores da velocidade da bola e da posição de cada robot em relação à mesma.

7.9.2.1 Ensaio 1 - Trajectória Rectilínea

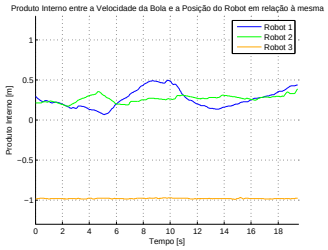
Nestes testes forçou-se a bola a movimentar-se segundo uma trajectória rectilínea a uma velocidade constante, variada entre 0.5 m/s e 1.5 m/s. Esta é a situação em que é mais simples para o controlador manter a geometria, dado que como a velocidade da bola é mantida com direcção e sentido constantes o controlador faz a previsão exacta da sua progressão.



(a) Plot XY



(b) Distância à Bola

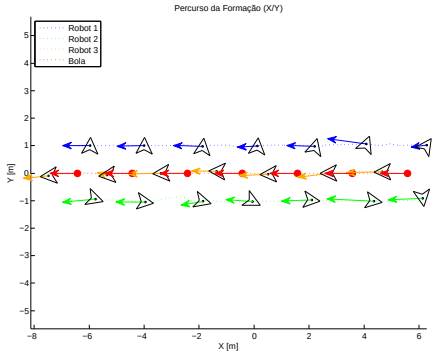


(c) Produto Interno de Versores

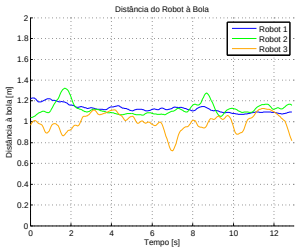
Figura 7.17: Manutenção da formação, ensaio 1 com $v_{ball} = 0.5m/s$

Para as velocidades da bola de 0.5 m/s e 1.0 m/s o desempenho do controlador é semelhante e apresenta uma precisão bastante aceitável. As distâncias à bola são mantidas muito próximas do valor desejado de 1 m enquanto que os valores dos produtos internos são também semelhantes aos pretendidos (0 para os robots 1 e 2 e -1 para o robot 3). É de notar que os gráficos de produto interno podem ser enganadores. O produto interno de dois versores, porque corresponde directamente ao coseno do ângulo entre estes (dado que têm módulo unitário e $\vec{v_1} \cdot \vec{v_2} = |\vec{v_1}||\vec{v_2}|cos(ang(\vec{v_1}, \vec{v_2}))$), varia muito mais rapidamente em torno de 0 do que em torno de -1. Assim, as variações maiores de produto interno verificadas para os robots 1 e 2 podem não significar necessariamente pior controlo (como aliás o mostram os plots XY feitos de cada ensaio).

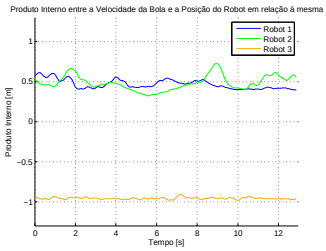
A $1.5m/s$ o controlador já demonstra uma certa dificuldade em manter a formação com precisão. Os robots 1 e 2, embora algo atrasados em relação à bola, conseguem manter uma distância razoavelmente próxima da pretendida. Já o robot 3 apresenta algumas dificuldades em acompanhar o percurso da bola, posicionando-se mais perto do que devia e oscilando a sua posição em torno da desejada. Ainda assim, a forma geométrica da formação é de um modo geral mantida.



(a) Plot XY

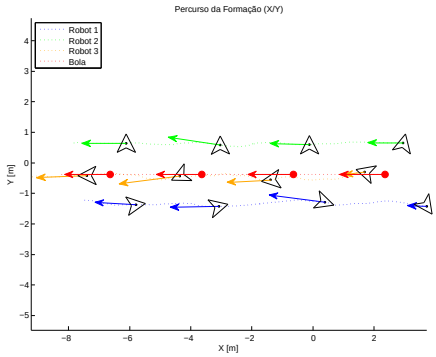


(b) Distância à Bola

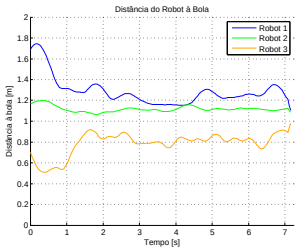


(c) Produto Interno de Versores

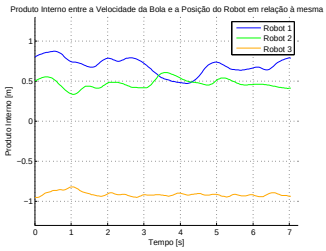
Figura 7.18: Manutenção da formação, ensaio 1 com $v_{ball} = 1.0m/s$



(a) Plot XY



(b) Distância à Bola

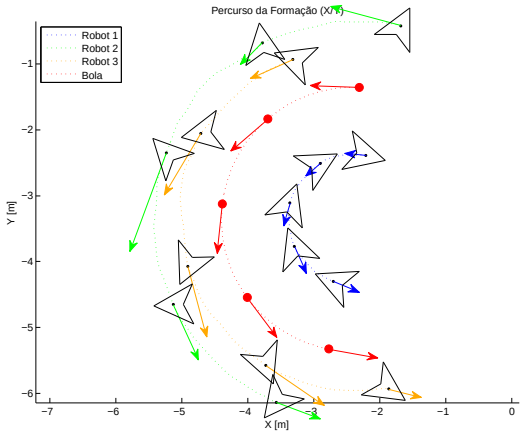


(c) Produto Interno de Versores

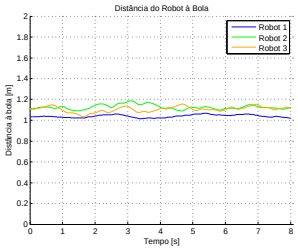
Figura 7.19: Manutenção da formação, ensaio 1 com $v_{ball} = 1.5m/s$

7.9.2.2 Ensaio 2 - Trajectória em Arco

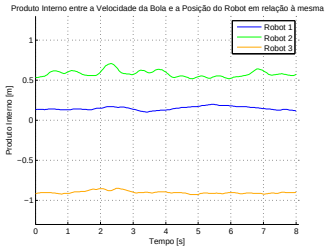
Neste ensaio foi testada uma trajectória da bola correspondente a um arco de circunferência com dois metros de raio. Ao invés do ensaio anterior, este é dos casos de mais difícil controlo dado que a direcção da velocidade da bola muda a cada instante e o controlador realiza a predição considerando que a velocidade se mantém constante ao longo de todo o horizonte de predição (o que é, obviamente, uma previsão errada).



(a) Plot XY



(b) Distância à Bola



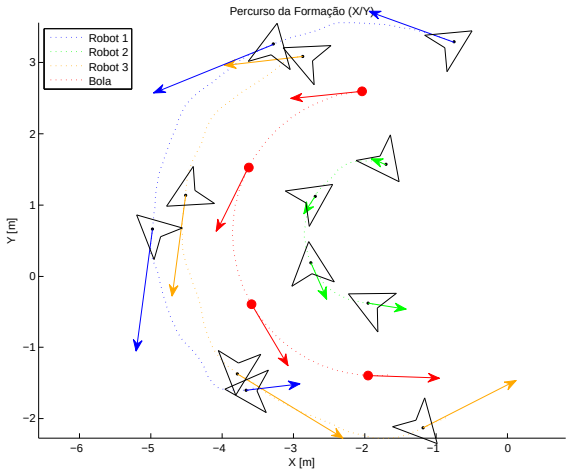
(c) Produto Interno de Versores

Figura 7.20: Manutenção da formação, ensaio 2 com $v_{ball} = 0.75 m/s$

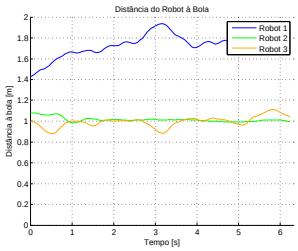
No primeiro caso de teste a bola deslocava-se com um módulo de velocidade de 0.75 m/s e no segundo com 1 m/s. Os resultados para $v_{ball} = 0.75 m/s$ encontram-se representados na figura 7.20. Conforme se pode observar, os três robots mantêm sem grandes desvios a distância de 1 m à bola. No que toca ao seu posicionamento em relação a esta há algumas particularidades a observar. Dos robots que devem estimar a velocidade da bola, o que está dentro do arco apresenta muito melhor posicionamento do que o que segue a bola do lado de fora da curva. Este resultado é de esperar, dado que para acompanhar o movimento da bola o robot de fora tem que se mover a uma velocidade consideravelmente superior à velocidade do robot de dentro (sendo esta tanto maior quanto a sua distância à bola e consequente raio de curvatura, pois $v = w \cdot r$). O robot que deve receber a bola apresenta alguns problemas de posicionamento devido ao facto da previsão que faz da evolução do estado da bola estar consideravelmente errada, conforme descrito no início desta secção.

Estes resultados são também uma oportunidade de destacar o carácter preditivo do controlador, facilmente observável na análise dos vectores velocidade da bola e do robot 3. A direcção do vector velocidade previsto pelo robot 3 é consistente com a posição para onde ele se deveria mover caso a velocidade da bola se mantivesse inalterada. Aqui está a explicação para os erros de posicionamento do robot 3.

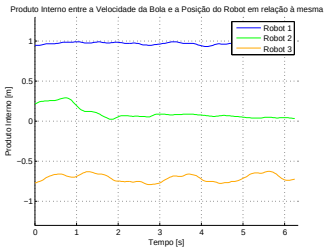
De seguida apresentam-se na figura 7.21 os resultados para um ensaio semelhante mas realizado com a velocidade da bola a 1 m/s.



(a) Plot XY



(b) Distância à Bola



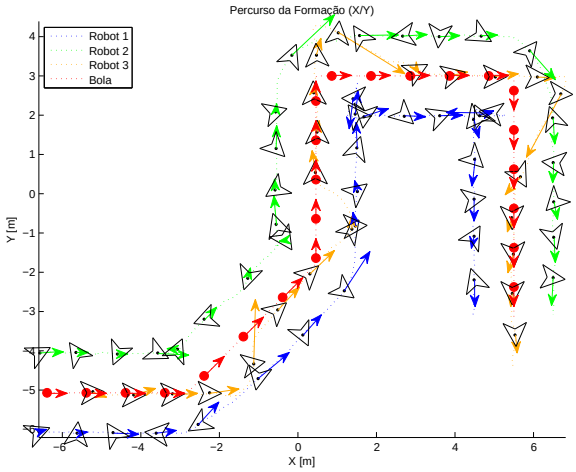
(c) Produto Interno de Versores

Figura 7.21: Manutenção da formação, ensaio 2 com $v_{ball} = 1.0m/s$

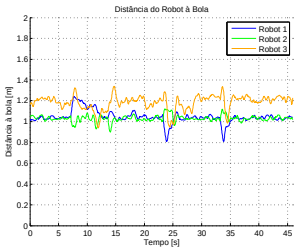
A degradação do desempenho do controlador, tendo em conta um aumento de apenas $0.25m/s$ na velocidade da bola, é imensa. O robot 1 consegue manter a distância e posição relativa à bola dentro de limites aceitáveis, mas os robots 2 e 3 apresentam um desempenho bastante fraco. Este ensaio demonstra uma das limitações do controlador: quando velocidades altas estão em jogo e no caso de estas mudarem de direcção constantemente, as predições erradas do controlador levam a uma quebra acentuada no desempenho no que toca à manutenção de uma formação em torno de um alvo.

7.9.2.3 Ensaio 3 - Cantos

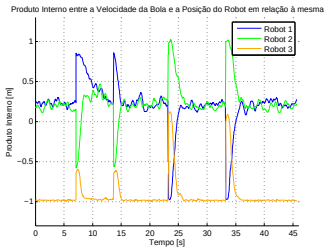
Este ensaio tem como objectivo averiguar o comoportamento do controlador quando a bola muda subitamente de direcção, nomeadamente se os robots são capazes de se reposicionar correctamente sem embaterem uns nos outros. Para esse efeito, dois tipos de cantos serão testados: um canto de 135 graus, menos abrupto, e um de 90 graus, recto. Os resultados apresentam-se na figura 7.22.



(a) Plot XY



(b) Distância à Bola

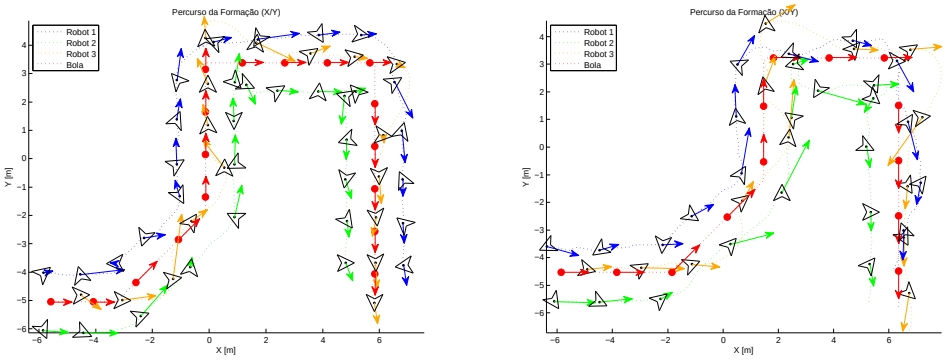


(c) Produto Interno de Versores

Figura 7.22: Manutenção da formação, ensaio 3 com $v_{ball} = 0.5m/s$

Os resultados obtidos são bastante animadores. A cada mudança de direcção a formação roda como um todo em torno da bola e os robots reposicionam-se de modo a ocupar as sua posições correctas. Observando os plots de distância e produto interno, a formação demora cerca de 3 segundos a reposicionar-se. Foram realizados dois ensaios semelhantes com velocidades da bola iguais a 0.75 m/s e 1 m/s, cujos resultados se apresentam na figura 7.23 sob a forma de plots XY.

Como seria de esperar, o aumento da velocidade da bola tem uma influência considerável na qualidade do resultados obtidos. Os robots continuam-se a conseguir reposicionar após o canto, mas demoram consideravelmente mais tempo e a geometria da formação como um todo está bastante mais disjunta.

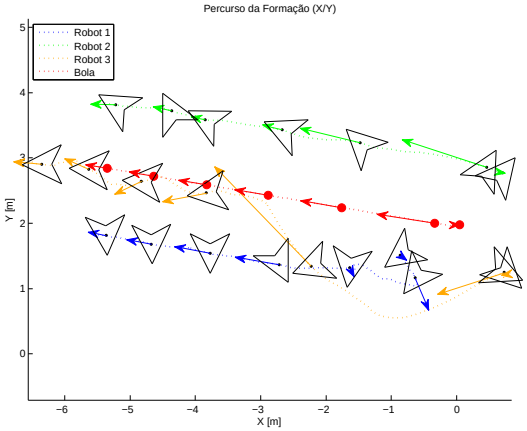


(a) Plot XY, $v_{ball} = 0.75m/s$ (b) Plot XY, $v_{ball} = 1.0m/s$

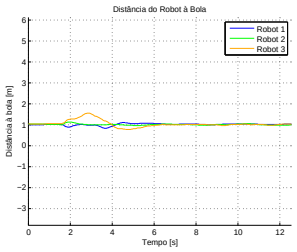
Figura 7.23: Manutenção da formação, ensaio 3 com diferentes velocidades

7.9.2.4 Ensaio 4 - Trajectórias livres

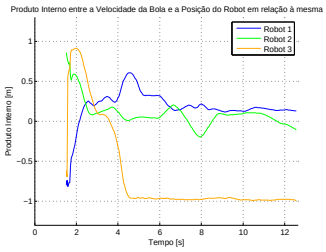
Finalmente realizaram-se vários ensaios deixando a bola circular livremente após a aplicação de uma força, com o intuito de averiguar o comportamento da formação face à desaceleração da bola devido às forças de atrito. Estes ensaios correspondem a situações reais que podem ocorrer durante um jogo, quando a bola é chutada por um robot ou ressalta em qualquer obstáculo.



(a) Plot XY



(b) Distância à Bola

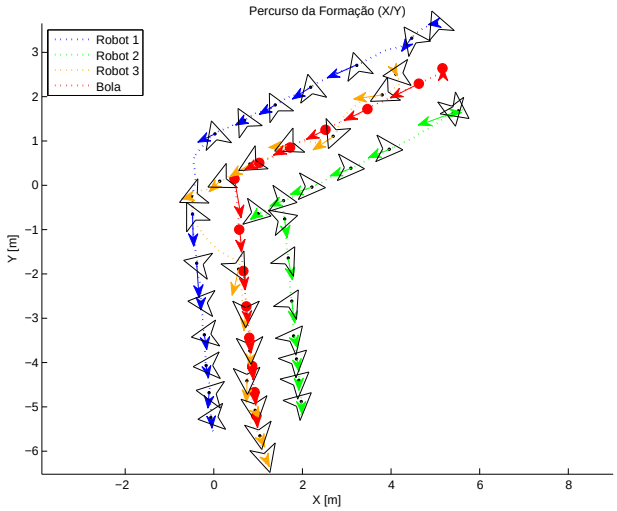


(c) Produto Interno de Versores

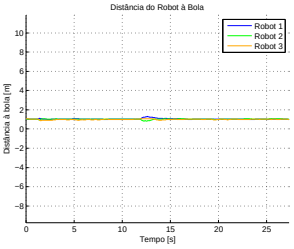
Figura 7.24: Manutenção da formação, ensaio 4 com um impulso na bola

O primeiro teste consiste na aplicação de um impulso inicial à bola fazendo-a rolar na direcção oposta à posição do robot 3, de modo que este tenha que se reposicionar. A bola encontrava-se inicialmente estacionária. Os resultados podem ser observados na figura 7.24. A análise dos resultados indica que a formação convergiu em cerca de 3 segundos, sendo mantida de forma consistente independentemente da desaceleração da bola.

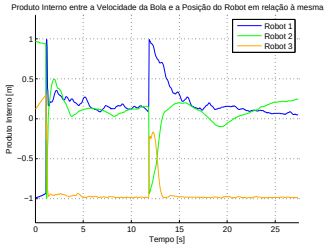
No segundo teste foram aplicados dois impulsos: um primeiro para iniciar o movimento da bola e o segundo para lhe mudar a direcção. Os resultados são descritos pela figura 7.25.



(a) Plot XY



(b) Distância à Bola

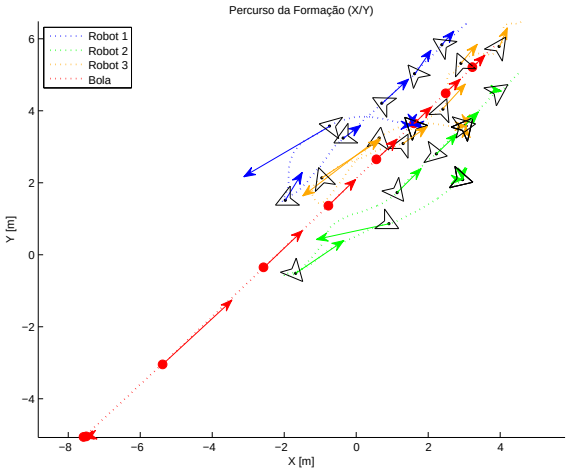


(c) Produto Interno de Versores

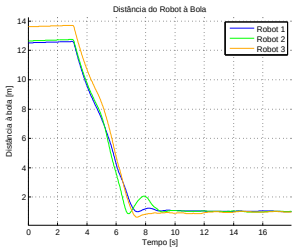
Figura 7.25: Manutenção da formação, ensaio 4 com dois impulsos na bola

O controlador reage bastante bem à mudança de velocidade e direcção no movimento da bola. O robot 3 reposiciona-se com uma rapidez considerável, em cerca de um segundo. A geometria da formação é retomada completamente em cerca de 5 segundos, com o posicionamento dos robots 1 e 2 nos locais correctos.

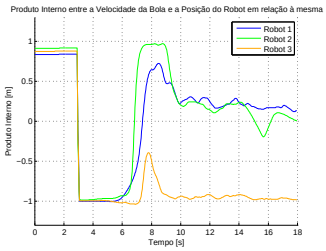
Nesta série de ensaios, um último teste foi realizado. Com os robots desactivados em posições próximas e aleatórias, a uma distância considerável da bola, foi dado um impulso à mesma para se mover na direcção geral dos robots. A ordem para movimento em formação foi então dada ao conjunto. Os resultados, algo confusos devido à sobreposição de trajectórias, são apresentados na figura 7.26.



(a) Plot XY



(b) Distância à Bola



(c) Produto Interno de Versores

Figura 7.26: Manutenção da formação, ensaio 4 com situação especial

Os robots começam por se movimentar na direcção da bola, de forma a encurtar a distância que os separa dela. O carácter preditivo do controlador permite-lhes aproximar-se dela e começar a acompanhar o movimento da bola quando a distância pretendida de 1 metro é atingida, evitando qualquer colisão entre eles ou com a bola. É de notar que pouco overshoot ou undershoot ocorre nas distâncias de cada robot em relação à bola. Quando esta distância é atingida os robots são capazes de se posicionar na sua posições ideais, convergindo a formação para acompanhar o movimento da bola com a geometria desejada. Este é uma situação complicada, na medida em que a proximidade entre os robots quando se movem para a bola e o facto de esta se estar a mover em direcção aos mesmos torna a ocorrência de colisões entre robots ou de um robot com a bola muito provável. O controlador mostrou ser capaz de evitar ambas.

Com este teste se concluíram os ensaios em ambiente de simulação. Os resultados obtidos permitiram analisar o desempenho do controlador sob diversas situações, determinando os seus pontos fortes e as suas limitações. Estas conclusões serão apresentadas na secção seguinte.

7.10 Conclusão

Foi descrito neste capítulo o projecto e implementação de um controlador de formações baseado em MPC, tomando como ponto de partida o controlador de trajectória descrito no capítulo anterior. A grande quantidade de ensaios em simulação permitiu a obtenção de dados muito pertinentes para a análise do comportamento do controlador. Infelizmente, e devido a uma série de circunstâncias, não foi possível realizar ensaios com os robots reais. Por um lado, e porque a equipa está em reformulação, só existia durante o desenvolvimento da tese um robot perfeitamente operacional. Por outro, o sistema de visão utilizado para obter dados de posição e orientação dos robots reais apenas permite cobrir uma área muito reduzida e insuficiente para ensaios de formações. O desenvolvimento de um novo sistema de aquisição de dados (porque o actual está limitado em área do campo coberta pela câmara pela altura em que esta pode ser montada no tecto do laboratório) quase que justificaria por si uma outra tese de mestrado.

A framework desenvolvida mostrou ser bastante flexível e facilmente adaptável, passando a implementação de novas formações pelo estudo da geometria da formação pretendida (ou dos critérios que se pretende manter em relação a um alvo um ou líder, pois a formação não tem que ser absolutamente rígida) e derivação das funções custo correspondentes para cada robot. A infra-estrutura de software da equipa 5DPO (nomeadamente, mDec + Coach) foi utilizada para implementar os controladores, tendo-se apresentado bastante estável.

O controlador projectado é capaz de fazer a equipa de robots convergir para a geometria pretendida em torno de um alvo, mesmo estando cada robot consideravelmente distante deste. O mérito deve ser atribuído ao algoritmo de minimização utilizado, RPROP, já que utilizando o Steepest Descent esta convergência não se verifica. Devido à grande penalização dada à proximidade extrema entre robots (utilizando uma função custo do tipo $f(x) = \frac{1}{x}$ para este elemento e dando-lhe um peso elevado no cálculo do custo completo) as colisões entre robots são efectivamente limitadas ou evitadas completamente.

No que toca à manutenção da formação os resultados obtidos são bastante dependentes das características da velocidade do alvo (a bola, neste caso). Para velocidades baixas os resultados são obviamente melhores, dado que os erros derivados de previsões incorrectas não resultam em desvios tão graves da geometria pretendida como quando as velocidades são elevadas. De um modo geral o controlador demonstrou reagir bastante bem a mudanças bruscas de direcção da velocidade do alvo, com a formação a rodar em torno da bola imediatamente após a mudança de direcção e a geometria a ser reconfigurada para acomodar a nova direcção. As dificuldades surgem quando velocidades altas do movimento do alvo são aliadas a variações frequentes de direcção ou sentido. O exemplo máximo ocorre numa trajectória circular da bola, caso em que a direcção da velocidade varia constantemente. Se considerarmos que o controlador opera com um ciclo de controlo de $40ms$ e utiliza um horizonte de predição de 10, a evolução da bola vai ser prevista em cada instante de controlo para os próximos 0.4 segundos. Para uma velocidade da bola de, assuma-se, 1.5 m/s o controlador vai prever um deslocamento em linha recta de 60 cm, o que será completamente diferente do arco de circunferência descrito pela bola durante esse instante (espe-

cialmente para circunferências de raios pequenos). Isto explica a dificuldade do controlador em manter a formação em trajectórias deste género. Felizmente, se considerarmos apenas situações que ocorram no futebol robótico, as trajectórias da bola serão essencialmente rectas com módulo de velocidade decrescente apresentando mudanças de direcção ocasionais devido a ressaltos ou remates de outros robots.

7.10.1 Trabalho futuro

Deixam-se ainda algumas possibilidades de melhoramentos ao controlador que poderiam ser um mais valia na resolução dos problemas descritos atrás. A sua implementação seria demasiado complexa e consequentemente demorada para poder ser feita no âmbito desta dissertação, pelo que ficam como sugestões de trabalho futuro.

Deixando de parte as limitações físicas do robot (limites dos motores, existência de fenómenos físicos como inércia e momentos de inércia, escorregamento das rodas), a grande limitação do controlador prende-se com o facto de existirem predições erradas em variadas situações. Estas podem ser consideradas de dois grandes tipos: erros na predição do movimento do alvo, e erros na predição do movimento dos outros robots. A única maneira de mitigar ambos resume-se a cada robot ter informações mais precisas do estado e intenções de cada robot e do alvo.

Predição do movimento dos robots - No que toca ao controlo da geometria de um robot em relação aos restantes, um grande melhoramento poderia ser introduzido ao ter cada robot a partilhar a sua previsão da evolução do seu estado para cada instante de controlo ao invés de apenas a sua posição e velocidade actuais. Isto permitiria aos restantes robots ter uma estimativa muito mais real da progressão dos colegas de equipa e assim melhorar o seu próprio controlo. Há no entanto a considerar que isto requer uma reestruturação profunda do sistema de comunicações. Uma vez que o protocolo UDP é utilizado, e sendo este um protocolo não determinístico, não é possível garantir que cada robot receberia estas informações no tempo correcto ou que sequer as receberia de todo. Como tal, teria que ser implementado um sistema de *timestamping* que mantivesse a coerência entre as previsões partilhadas por cada robot e o instante a que correspondem. Para mais, um sistema hierárquico de algum tipo teria que ser implementado, pois todos os robots não poderiam ao mesmo tempo partilhar a evolução do seu estado e calculá-la baseada nas evoluções partilhadas pelos outros.

Predição do movimento do alvo - Porque o alvo é um elemento estranho à formação, na medida em que não está em comunicação com nenhum dos robots, nunca é possível obter uma informação exacta da evolução do seu estado. No entanto, através da análise do histórico da sua velocidade (direcção, sentido, e módulo) pode ser possível estimar certas características da sua evolução futura. Tome-se o exemplo de uma bola que segue em linha recta, sujeita apenas ao atrito. Da análise da sua velocidade ao longo do tempo pode-se estimar a sua desaceleração e ter esta em conta quando se faz a previsão da evolução do estado da bola, ao invés de considerar a sua velocidade constante. O mesmo se aplica a um alvo que se mova em arco, podendo ser estimada a variação da direcção

da velocidade através do seu histórico. No fundo trata-se de acrescentar uma segunda camada de predição ao controlador, fazendo uma previsão da evolução da velocidade do alvo baseada no seu histórico e utilizar esta informação para prever a evolução da sua posição. É de notar que este método é altamente falível para alvos com motorização própria (veículos autónomos ou controlados por um operador humano), mas para alvos sujeitos apenas a forças naturais (atritos, gravidade, e afins) poderia significar um grande melhoramento.

Capítulo 8

Conclusão

Foram cumpridos no decurso desta dissertação todos os objectivos propostos inicialmente, pelo que o balanço final do projecto é claramente positivo. A divisão do trabalho em dois grandes blocos (controlador de trajectória e controlador de formações) revelou-se bastante benéfica para o progresso suave to projecto como um todo na medida em que quando se iniciou o trabalho no controlo de formações já existia uma infra-estrutura de controlo preditivo testada e adequadamente parametrizada ao robot em questão para utilizar como base. Este capítulo sistematiza e analisa criticamente o trabalho produzido, identificando as principais contribuições e objectivos atingidos e finalizando com sugestões de trabalho futuro.

8.1 Comentários gerais

No que toca ao controlo de trajectórias a principal novidade em relação às técnicas estudadas previamente traduz-se na utilização de um modelo dinâmico simplificado para o controlador preditivo, o que reduz largamente o peso computacional da previsão executada em cada ciclo de controlo permitindo a utilização de horizontes de controlo e predição elevados mesmo em máquinas de especificações relativamente modestas (como os portáteis utilizados nos robots da equipa 5DPO). Este método revelou-se capaz de fornecer bons resultados no seguimento de trajectórias com mudanças bruscas de direcção e orientação, a velocidades elevadas. Requer no entanto a adequação prévia dos parâmetros do modelo ao robot em questão, o que só pode ser conseguido com uma grande quantidade de ensaios.

Os resultados obtidos para o controlo de formações vão de encontro aos requisitos do projecto da *FCT* a que esta tese está afecta, descrito no capítulo 1. A principal contribuição para este projecto centra-se no projecto e implementação de uma framework descentralizada para controlo de formações baseada em controlo preditivo, onde a geometria da formação não é definida previamente mas advém da minimização de funções custo talhadas para optimizar certas características da mesma. Embora o controlador seja distribuído e cada robot resolva o seu próprio problema de

optimização, o facto de as funções custo estarem acopladas (por partilha de informações do estado de cada robot entre os membros da equipa e utilização destes dados nas funções custo e previsões) explica a consistência da formação e a capacidade de evitar colisões. A framework implementada fornece uma base sólida para a continuação do desenvolvimento de métodos descentralizados de controlo de formações de robots móveis, podendo servir de ponto de partida para soluções mais complexas.

Há também a destacar o papel essencial do *SimTwo* no decurso do trabalho. Sem este motor de simulação não teria sido possível o enorme volume de ensaios realizado (largamente superior a duas centenas) para optimização do controlador e averiguação das suas capacidades. Muito menos teriam sido possíveis os ensaios do controlador de formações, por falta de robots funcionais em número suficiente e de um sistema capaz de capturar os dados necessários. A utilização do SimTwo permite também isolar os problemas de controlo dos problemas de localização dos robots ou da bola, fornecendo efectivamente uma boa medida do desempenho do controlador.

Ambos os controladores foram projectados e implementados tendo em conta a sua provável utilização no futebol robótico, pelo que se pretende que o trabalho aqui desenvolvido possa vir a ser directamente aplicado na equipa que lhe serviu de testbed, 5DPO.

8.2 Objectivos atingidos

Os objectivos finais cumpridos vão de encontro aos propostos inicialmente. Assim, e sistematizando, as seguintes metas foram atingidas:

- Estudo do estado da arte no que toca a controlo de alto desempenho de seguimento trajectórias, com sistematização e apresentação das principais técnicas utilizadas actualmente, dando-se destaque a controladores do tipo preditivo. Muitas das técnicas estudadas foram utilizadas directamente ou adaptadas para utilização nos algoritmos produzidos no âmbito desta dissertação.

- Estudo do estado da arte do controlo de formações de veículos móveis, incluindo a recolha das técnicas desenvolvidas recentemente e a adaptação de algumas destas ideias ao projecto em causa. Também aqui foi dado destaque às tecnologias baseadas em controlo preditivo.

- Projecto de um controlador preditivo para seguimento de trajectórias utilizando um modelo dinâmico simplificado do robot utilizado, o que se traduz na possibilidade de utilização de horizontes de predição e controlo maiores.

- Implementação da arquitectura de controlo proposta em Lazarus/FreePascal utilizando como base o software mDec da equipa de futebol robótico 5DPO.

- Validação do controlador projectado por meio de testes intensivos em simulação e com o robot real, assim como adequação dos parâmetros do modelo utilizado no controlador ao robot utilizado em simulação.

- Projecto e desenvolvimento de uma arquitectura descentralizada baseada em controlo preditivo para controlo de formações de robots móveis. Esta consiste numa framework flexível que facilita a implementação de novas formações definidas através das funções custo empregues por cada robot.

- Implementação dos controladores de formação projectados sobre o software mDec e Coach, utilizados para controlo da equipa 5DPO.

- Estudo e implementação nos controladores desenvolvidos de uma formação directamente ligada ao futebol robótico, destinada a optimizar a percepção da velocidade da bola e recepção da mesma por uma equipa de 3 robots.

- Realização de ensaios extensivos do controlador de formações em simulação de modo a validar o seu funcionamento e averiguar o seu desempenho sob diversas situações.

- Identificação das principais qualidades e limitações dos algoritmos produzidos, assim como sugestão de possíveis avanços a implementar para as ultrapassar.

8.3 Trabalho futuro

O trabalho aqui realizado apresenta potencial para ser posteriormente desenvolvido e melhorado, tanto nas vertentes de controlo de trajectória como de controlo de formação. Apresentam-se de seguida algumas sugestões de continuação do projecto.

8.3.1 Controlador de Trajectória

A continuação do desenvolvimento do controlador de trajectória, nas suas duas vias mais óbvias, centra-se por um lado na alteração do modelo utilizado para as previsões e por outro no desenvolvimento e teste de outros algoritmos de optimização. Seria interessante verificar o quão realista se poderia tornar o modelo sem comprometer a capacidade de utilizar horizontes de predição e controlo elevados. Um bom ponto de partida seria começar por implementar e parametrizar o modelo do controlador PID + motor associado a cada roda para substituir o sistema LTI de primeira ordem utilizado. A utilização do método de Euler ou de um outro método numérico para resolver as equações diferenciais resultantes poderia fornecer uma simulação em tempo útil que fosse bastante mais realista que o modelo utilizado actualmente. Seria também de valor testar outros algoritmos de optimização para além do *Steepest Descent* e do RPROP (ou testar mais extensivamente este último), o que não houve tempo para fazer convenientemente no decurso da dissertação. Finalmente, os mesmo ensaios que foram realizados em simulação para determinação dos parâmetros do modelo (a constante de tempo do sistema de primeira ordem que modela a resposta dos motores) deveriam ser realizados com o robot real de maneira a validar os parâmetros já obtidos ou a determinar os correctos para o robot real.

8.3.2 Controlador de Formações

No que toca ao controlador de formação algumas das melhorias possíveis foram já discutidas na secção 7.10.1. A incorporação destas seria definitivamente um mais valia para o melhoramento do desempenho do controlador. Há ainda um aspecto muito importante a considerar: a validação do controlador utilizando equipas de robots reais. As simulações realizadas são, como foi frisado várias vezes ao longo desta dissertação, bastante realistas. No entanto não têm em conta determinados erros que ocorrem com a utilização de robots reais (erros de localização de cada robot, atrasos na comunicação entre supervisor e equipa que podem levar a que cada robot não tenha as posições correctas dos companheiros de equipa, erros da determinação da posição da bola). Porque estes ensaios com robots reais não foram possíveis, pelas razões apontadas anteriormente, seriam o próximo passo lógico no desenvolvimento do controlador de formações.

A formação implementada poderia também ser de certa forma melhorada. Conforme referido atrás, a geometria da formação não está definida previamente, resultando sim da optimização da função custo em cada robot. No entanto as distâncias à bola desejadas são impostas, pré-determinadas. O mesmo acontece para a posição dos robots em relação à bola. Idealmente deveria ser determinada uma função que relacionasse directamente as características da bola conforme observada pelo robot (posição na imagem, número de píxeis na imagem correspondentes à bola) com a qualidade da estimação da velocidade. Da optimização desta função custo (em conjunto com os elementos responsáveis por evitar colisões entre robots) resultariam as distâncias e posições ideias. Há que considerar que isto requereria mais uma camada de simulação (responsável por determinar como é que movimentos do robot afectariam as características da bola conforme observada na imagem). Se a complexidade acrescida na simulação justificaria os ganhos em termos de precisão do controlo só a implementação deste sistema o diria.

Finalmente, a implementação de novas formações (cujo processo é bastante fácil uma vez determinadas as funções custo, dado o método como o controlador foi programado) seria interessante. Com a framework desenvolvida torna-se muito simples implementar formações geométricas fixas baseadas em seguimento de líder, formações para seguimento de alvos móveis, e afins.

Referências

[1] André Gustavo Scolari Conceição. *Controlo e Cooperação de Robôs Móveis Autónomos Omnidireccionais*. Tese de Doutoramento em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto - Portugal, 2007.

[2] Fontes Dalila B.M.M. Fontes, Fernando A.A.C e Amélia C.D. Caldeira. Behavior-based formation control for multirobot teams. *Optimization and Cooperative Control Strategies*, páginas 371 – 384, 2009.

[3] Heonyoung Lim, Yeonsik Kang, Jongwon Kim, e Changwhan Kim. Formation control of leader following unmanned ground vehicles using nonlinear model predictive control. Em *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, páginas 945 –950, 14-17 2009.

[4] Rolf Findeisen e Frank Allgöwer. An introduction to nonlinear model predictive control. *21st Benelux Meeting on Systems and Control*, 2002.

[5] Paulo J. Costa. Simtwo, 2010. Available from http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo,.

[6] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[7] J. L. Martins de Carvalho. *Dynamical systems and automatic control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[8] P. Muir e C. Neuman. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. volume 4, páginas 1772 – 1778, mar 1987.

[9] Yamato S. Spyros G.T Jun T. Keigo, W. e F. Toshio. Feedback control of an omnidirectional autonomous platform for mobile service robots. *Journal of Intelligent Robotic Systems*, 1998.

[10] T.-H.S. Li, Sheng-Sung Jian, e Ming-Che Tsai. Design and implementation of fuzzy trajectory following and planning control for mobile robots. volume 1, páginas 453 –458 vol.1, 2001.

[11] André Scolari Conceição, A. Paulo Moreira, e Paulo J. Costa. Controller optimization and modelling of an omni-directional mobile robot. *CONTROLO'2006 - The 7th Portuguese Conference on Automatic Control,11-13 September 2006, Lisbon-Portugal*, 2006.

[12] André Scolari Conceição, A. Paulo Moreira, e Paulo J. Costa. Trajectory tracking for omni-directional mobile robots based on restrictions of the motor's velocities. *8th International IFAC Symposium on Robot Control - Syroco 06, Santa Cristina Convent, University of Bologna (Italy), September 6 - 8*, 2006.

[13] Pedro Costa, Paulo Moreira, e Paulo Costa. Real-time path planning using a modified a* algorithm. *ROBOTICA 2009 - 9th Conference on Mobile Robots and Competitions, 7th May, Castelo Branco, Portugal*, 2009.

[14] M. Egerstedt, X. Hu, e A. Stotsky. Control of mobile platforms using a virtual vehicle approach. *IEEE transactions on Automatic Control, Vol. 46, No. 11*, 2001.

[15] Tiago Pereira do Nascimento. *Controle de Trajetória de Robots Móveis Omni-Direccionais: Uma Abordagem Multivariável*. Tese de Mestrado em Engenharia Electrotécnica e de Computadores, Universidade Fereral da Bahia, Escola Politécnica, 2009.

[16] E. F. Camacho e C. Bordons. *Model Predictive Control*. Springer, segunda edição, 2004.

[17] Kiattisin Kanjanawanushkul e Andreas Zell. Path following for and omnidirectional mobile robot based on model predictive control. *2009 IEEE International Conference on Robotics and Automation*, 2009.

[18] Stavros G. Vougioukas. Reactive trajectory tracking for mobile robots based on non linear model predictive control. *2007 IEEE International Conference on Robotics and Automation*, 2007.

[19] Gregor Klancar e Igor Skrjanc. Predictive trajectory tracking control for mobile robots. *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*, páginas 373 –378, 30 2006-sept. 1 2006.

[20] Dongbing Gu e Huosheng Hu. Wavelet neural network based predictive control for mobile robots. volume 5, páginas 3544 –3549 vol.5, 2000.

[21] D.R. Ramirez, D. Limon, J. Gomez-Ortega, e E.F. Camacho. Nonlinear mbpc for mobile robot navigation using genetic algorithms. volume 3, páginas 2452 –2457 vol.3, 1999.

[22] X. Jiang, Y. Motai, e X. Zhu. Predictive fuzzy logic controller for trajectory tracking of a mobile robot. páginas 29 – 32, june 2005.

[23] André Scolari Conceição, A. Paulo Moreira, e Paulo J. Costa. A nonlinear model predictive control strategy for trajectory tracking of a four-wheeled omnidirectional mobile robot. *Proceedings of 2007 IEEE International Symposium on Industrial Electronics, pag. 2161-2165, Centro Cultural and Centro Social Caixanova - Vigo, Spain, June 4-7*, 2007.

[24] André Scolari Conceição, A. Paulo Moreira, Hélder P. Oliveira, A. Sousa e Silva, e Diogo Oliveira. A nonlinear model predictive control of an omni-directional mobile robot. *, Proceedings of 2007 IEEE International Symposium on Industrial Electronics, pag. 2161-2165, Centro Cultural and Centro Social Caixanova - Vigo, Spain, June 4-7*, 2007.

[25] R. Fletcher. *Practical Methods of Optimization*. Wiley, segunda edição, 2001.

[26] R. Ghabcheloo, A. Pascoal, C. Silvestre, e I. Kaminer. Coordinated path following control of multiple wheeled robots with directed communication links. Em *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, páginas 7084 – 7089, 12-15 2005.

[27] K. Kanjanawanishkul e A. Zell. A model-predictive approach to formation control of omnidirectional mobile robots. Em *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, páginas 2771 –2776, 22-26 2008.

[28] Kar-Han Tan e M.A. Lewis. Virtual structures for high-precision cooperative mobile robotic control. Em *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 1, páginas 132 –139 vol.1, 4-8 1996.

[29] Wei Ren e R.W. Beard. A decentralized scheme for spacecraft formation flying via the virtual structure approach. Em *American Control Conference, 2003. Proceedings of the 2003*, volume 2, páginas 1746 – 1751, june 2003.

[30] T. Balch e R.C. Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926 –939, dec 1998.

[31] K. Kanjanawanishkul e A. Zell. Distributed model predictive control for coordinated path following control of omnidirectional mobile robots. Em *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, páginas 3120 –3125, 12-15 2008.

[32] R. Ghabcheloo, A. Pascoal, C. Silvestre, e I. Kaminer. Coordinated path following control of multiple wheeled robots with directed communication links. Em *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, páginas 7084 – 7089, 12-15 2005.

[33] W.B. Dunbar e R.M. Murray. Model predictive control of coordinated multi-vehicle formations. Em *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 4, páginas 4631 – 4636 vol.4, 10-13 2002.

[34] Robocup. http://www.robocup.org.

[35] Manuel Carlos Monteiro Gouveia. *Estudo e implementação de um algoritmo de localização baseado em correspondência de mapas*. Tese de Mestrado em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto - Portugal, 2008.

[36] Hélder P. Oliveira, Armando J. Sousa, A. Paulo Moreira, e Paulo J. Costa. Dynamical models for omni-directional robots with 3 and 4 wheels. *ICINCO 2008 -International Conference on Informatics in Control, Automation and Robotics. ISBN: 978-989-8111-31-9.Vol I. pp.189-196. Funchal, Madeira, Portugal. May 11–15*, 2008.

[37] Hélder P. Oliveira, Armando J. Sousa, A. Paulo Moreira, e Paulo J. Costa. Precise modeling of a four wheeled omni-directional robot. *Encontro Cientifico do ROBOTICA2008, pag. 57-62, Aveiro, Portugal, 2 a 6 de Abril*, 2008.

[38] S.J. Qin e T.A. Badgwell. An overview of industrial model predictive control technology. Em *Fifth International Conference on Chemical Process Control - CPC V*, páginas 232 – 256, 1996.

[39] S.J. Qin e T.A. Badgwell. An overview of model predictive control applications. Em *Nonlinear predictive control*, páginas 369 – 393, 2000.

[40] Fernando A. C. C. Fontes. A general framework to design stabilizing nonlinear model predictive controllers.

[41] D. Q. Mayne, J. B. Rawlings, C. V. Rao, e P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica 36, 789-814*, 1999.

[42] Digital transfer functions for microcomputer control. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(12):856 –860, dec. 1979.

[43] Russel Smith. Open dynamics engine, 2010. Available from http://www.ode.org.

[44] Renato Miguel dos Santos Caldas. *Modelação e Simulação de um Robot Omnidireccional de 3 rodas*. Tese de Mestrado em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto - Portugal, 2009.

[45] M. Riedmiller e H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. Em *Neural Networks, 1993., IEEE International Conference on*, páginas 586 –591 vol.1, 1993.