FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Movimentação e Ensinamento de um Robô NAO

Eduardo Moreira da Mota

Mestrado Integrado em Engenharia Electrotécnica e Computadores

Orientador: Prof. Dr. António Paulo Gomes Mendes Moreira

Co-orientador: Tiago Pereira do Nascimento

Julho de 2011

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2010/2011

A Dissertação intitulada

"Movimentação e Ensinamento de um Robô NAO"

foi aprovada em provas realizadas em 18-07-2011

o júri

Presidente Professor Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Luís Sousa de Magalhães Lima
Professor Adjunto Departamento Electrotécnia da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança

Professor Doutor António Paulo Gomes Mendes Moreira
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Eduardo Moreira da Mota

Faculdade de Engenharia da Universidade do Porto

Resumo

Este documento apresenta a implementação de um sistema de teleoperação dos braços de um robô NAO, utilizando a câmara Kinect da Microsoft como um meio de ensinar o robô através da demonstração de movimentos. O conceito permite propor um método inovador que possibilita um controlo com elevada liberdade de manipulação, utilizando tecnologia completamente não invasiva, através de uma câmara capaz de captar não só a cor como também a profundidade de cada píxel no seu plano de captura. Este documento descreve o processo de aquisição de imagens, processamento e geração de coordenadas das articulações do corpo humano, utilizadas para determinar os ângulos de referência para o robô. Devido ao ruído de alta frequência da câmara, foi utilizado um filtro do tipo passa-baixo sobre a informação devolvida por esta, cuja configuração e resultados se encontram também descritos.

Finda a determinação dos ângulos, estas são agregadas numa mensagem que é enviada via *wireless* para o robô que, efectuando o seu desagregamento, os reconhece e os aplica nas respectivas articulações.

É descrito ainda um método que permite determinar o conjunto de actuações que, num manipulador, coloca a extremidade numa posição desejada. Este é um processo baseado em análise numérica, válido para qualquer configuração do manipulador. Este método é particularmente interessante quando o número de articulações a determinar supera o número de equações linearmente independentes, como acontece com as pernas e braços do NAO.

Abstract

This document presents the implementation of a teleoperation system of the arms of a NAO robot using the Kinect camera from Microsoft as a way to teach the robot by demonstration of preset movements. The concept proposes an inovative method that turns possible a control with high manipulation freedom, using completely non-invasive technology, by the means of a camera that is able to capture not only the color, but also the depth of each pixel in its frame. This document describes the process of image aquisition, processment and generation of the coordinates of each joint of the human body, used to determine the reference angles for the robot. Due to the existence of high-frequency noise in the camera, it was used a low-pass filter on the information it returns, which configuration and results are also described.

After the determination of the angles, these are assembled in a message, which is sent wirelessly to the robot that, by performing its disassemblement, recognizes them, and applies them in the respective joints.

It is also described a method that allows the computation of a group of actuations that, in a manipulator, puts the end effector point in a desired position. This process is based on mathematical analysis, valid to every configuration of the manipulator. This method is specially interesting when the number of unknown joints surpasses the number of linearly independent equations, which happens with both the legs and arms of the NAO.

Agradecimentos

Um curso superior não se faz sozinho. Foram muitas pessoas que me apoiaram, e tornaram possível atingir este marco importante na minha vida. No entanto, gostaria de não deixar de evidenciar algumas.

Em primeiro lugar, gostaria de agradecer o apoio imprescindível da minha família, por reconhecerem e apoiarem a minha ambição para concluir um curso superior. Teço um agradecimento muito especial à minha mãe, pelo amor e espírito de sacrifício constante, para que nada me faltasse, e pudesse progredir no curso sem outras preocupações.

Agradeço aos meus orientadores, o professor António Paulo Moreira e o Tiago Nascimento, pela confiança que depositaram em mim, acreditando no sucesso deste projecto, pela amizade e os muitos e imprescindíveis conselhos; e também a todos os meus colegas do laboratório de Robótica e Sistemas Inteligentes, e ao Nima Shaffi, pelas dicas, sugestões e ideias que, aqui e ali, fizeram do meu projecto de dissertação um projecto não só interessante, como também divertido.

Ainda, gostaria de reconhecer o apoio dos meus amigos, pelos momentos de descontracção, pelo apoio incondicional, pelas risadas, mas sobretudo pelo sentimento genuíno de amizade. Felizmente são muitas as pessoas, mas quero destacar a Mara Antunes, a Maria Kravtsova, o Daniel Magalhães e a Vivian Abram. Também, destaco e agradeço ao Samuel Cunha, pelos cafés *am Strand*, pelos infinitos cafés na máquina do DEEC, pelo companheirismo e "brincadeira-parvismo"constantes.

Agradeço profundamente à professora Paula Rebocho, que desde cedo me incutiu o gosto pelo estudo, e acima do tudo por me ter feito sentir desde cedo o desejo de progredir na minha vida académica.

Por fim, agradeço à Graça, pela tolerância, apoio e carinho sempre presentes. Por me fazer saber que seria capaz de chegar tão longe, e por me incentivar a não desistir às primeiras dificuldades que se colocaram ao longo da dissertação, e também ao longo do curso. Por reconhecer o esforço e motivação com que encarei a tese, e por ter sabido aguentar pacientemente os meus longos dias de trabalho.

Eduardo Mota

"Discipline is the bridge between goals and accomplishments."

Jim Rohn

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

3D-LIPM	*Three-Dimensional Linear Inverted Pendulum Mode*
API	*Application Programming Interface*
CoP	*Center of Pressure*
CPG	*Central Pattern Generator*
DCM	*Device Control Manager*
DEEC	Departamento de Engenharia Electrotécnica e Computadores
FEUP	Faculdade de Engenharia da Universidade do Porto
FPB	Filtro Passa-Baixo
GAOFSF	*Genetic Algorithm Optimized Fourier Series Formulation*
HSI	*Hue-Saturation-Intensity*
IEETA	Instituto de Engenharia Electrónica e Telemática de Aveiro
INESC	Instituto de Engenharia de Sistemas e Computadores
IMC	Índice de Massa Corporal
IP	*Internet Protocol*
RGB	*Red-Green-Blue*
SDRAM	*Synchronous Dynamic Random Access Memory*
TCP	*Transmission Control Protocol*
TFS	*Truncated Fourier Series*
UDP	*User Datagram Protocol*
ZMP	*Zero Moment Point*

Lista de símbolos

ω	Frequência angular
Ω	Deslocamento angular
g	Aceleração da gravidade
I	Momento de inércia
τ	Binário
f	Frequência linear
T	Período

Capítulo 1

Introdução

A ideia de criações semelhantes ao ser humano que o pudessem assistir é desde sempre uma área fascinante, pois permitiria ao Homem poder dedicar-se a outras tarefas, como a política, negócios, ou mesmo lazer. A primeira referência remete para 270 anos antes de Cristo, quando o antigo engenheiro grego Ctesibus criou órgãos e relógios de água com figuras móveis (1). Em 1921, o escritor checo Karel Capek utilizou pela primeira vez o termo *Robot*, na sua peça *Rossum's Universal Robots*.

Os primeiros projectos de máquinas capazes de desempenhar estas funções e que aproximaram o ser humano desta realidade remontam a 1495, quando Leonardo Da Vinci apresentou os planos para a construção daquele que viria a ser o primeiro robô humanóide capaz de realizar movimentos humanos como sentar, levantar e mover os braços, mantendo a ergonomia característica do ser humano.

No entanto, foi em 1939 que Isaac Asimov, na primeira das trinta e uma histórias de ficção científica da colecção *The Complete Robot* que escreveu, sugeriu que seres humanos e robôs poderiam não só coexistir no planeta, mas também interagir de forma inteligente entre eles, prevendo o nascimento da robótica como uma indústria promissora e de um enorme potencial. Mas é em 1942 que, na sua história *Runaround*, enunciou as três leis dos robôs, que se perpetuaram até aos dias de hoje:

- Um robô não deve ferir um ser humano ou, por inactividade, permitir que um ser humano seja ferido.
- Um robô deve obedecer às ordens de um ser humano, excepto se tais ordens implicarem ferir outros seres humanos.
- Um robô deve sempre proteger a sua própria existência, excepto se tal implicar desobedecer ou ferir seres humanos.

Na construção de robôs autónomos é importante considerar uma panóplia de factores cruciais ao seu desempenho, nomeadamente:

- Autonomia/eficiência energética
- Processamento
- Dimensões físicas
- Complexidade de hardware
- Programabilidade
- Estabilidade
- Precisão Sensorial
- Custos

De acordo com Gouaillier (2), a relação entre a altura e a massa de um robô fornece à partida uma boa estimativa do seu desempenho energético. Com efeito, um robô mais leve consegue produzir o mesmo movimento que um robô mais pesado, consumindo menos energia; os motores podem ser mais pequenos, e as perdas sob a forma de calor serão muito menos significativas. Robôs pequenos, além de uma melhor dinâmica, são tipicamente menos perigosos para o ser humano, e naturalmente mais baratos. A tabela 1.1 estabelece uma comparação entre alguns robôs humanóides funcionais, em função das suas massas, alturas, IMCs[1] e preços.

Tabela 1.1: Relação IMC/Preço entre robôs humanóides funcionais

Robô	**Massa (Kg)**	**Altura (m)**	**IMC**	**Preço**
KHR-2HV	1.3	0.34	10.9	1 K$
HOAP	7.0	0.50	28.0	50 K$
NAO	**4.5**	**0.57**	**13.5**	**10 KEuros**
QRIO	6.5	0.58	19.0	N/A
ASIMO	54.0	1.30	32.0	N/A
REEM-A	40.0	1.40	20.4	N/A
HRP-2	58.0	1.54	24.5	400 K$

1.1 O Robô NAO

NAO (pronunciado "nau") foi o nome dado ao robô humanóide produzido em 2005 pela empresa francesa Aldebaran Robotics, tendo como principal motivação ao seu desenvolvimento a necessidade de fornecer ao mercado da investigação e ensino um robô humanóide a um preço mais reduzido que os robôs até à data existentes. De facto, a criação deste robô impulsionou não só o estudo da locomoção humanóide no meio académico como também de todas as particularidades da locomoção bípede, baseadas no processamento de imagens e ultra-sons, entre outras.

[1] Índice de massa corporal. Este valor é obtido dividindo a massa de um corpo pelo quadrado da sua altura.

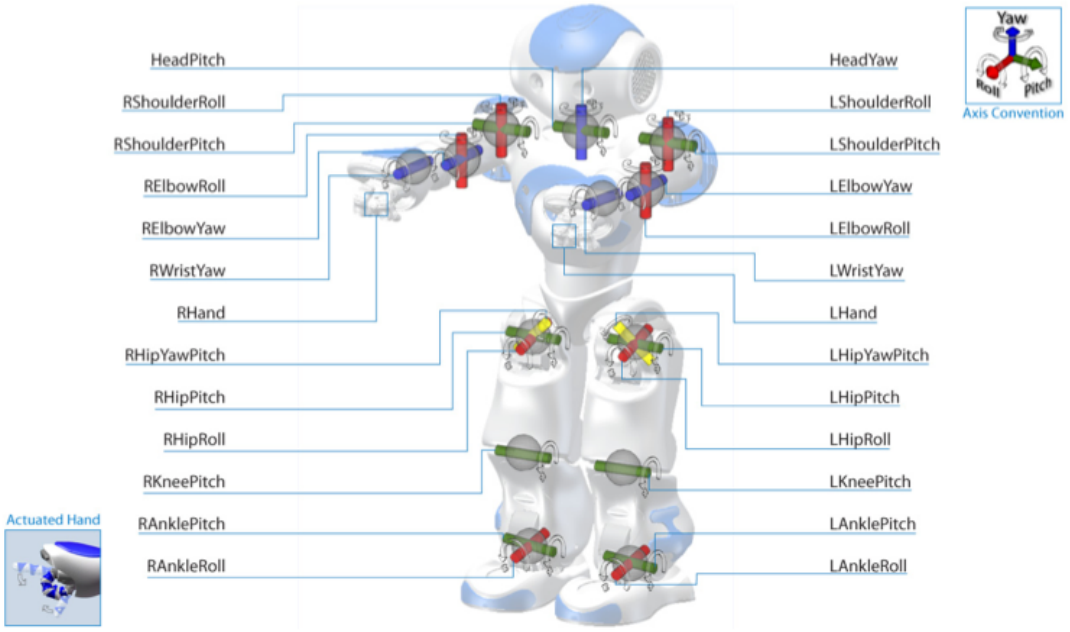

Figura 1.1: Ligações e graus de liberdade do NAO (4)

Com sensivelmente 58 centímetros de altura, este robô foi projectado com o compromisso de responder de forma rápida a um variado conjunto de estímulos, oferendo igualmente um vasto conjunto de tarefas que este poderia executar, desde andar até apanhar objectos, ou mesmo recuperar autónoma e automaticamente de quedas (3). Por isso, a sua constituição consiste num total de 17 elos de ligação, unidos por um total de 13 articulações rotativas. Esta configuração (figura 1.1) oferece um total de 25 graus de liberdade de manipulação.

1.2 A Liga RoboCup

Guy Richards, conhecido jogador de futebol, disse: *"Se a robótica continuar a evoluir à velocidade actual, em 2050 todos poderemos assistir a um Campeonato do Mundo jogado entre humanos e robôs"* (5). Com efeito, este é o objectivo que moveu à criação e expansão da liga internacional de futebol robótico *RoboCup*. O conceito, ainda embrionário, surge em 1993, por parte de um grupo de investigadores japoneses, mas que depressa se estendeu para lá das fronteiras. Este consiste numa competição jogada unicamente entre robôs, sem qualquer auxilio humano e obedecendo a um conjunto de regras estandardizadas, constituindo um meio de promover e desafiar a investigação na área da inteligência artificial, e outras relacionadas (6).

A criação do NAO em 2005 constituiu um marco histórico na história do *RoboCup*, levando à sua adopção como plataforma oficial da prova, por parte da comissão organizadora deste campeonato, em 2007, destronando o robô-cão *Aibo*, da Sony (7). Pela primeira vez na história da competição foi adoptado um robô humanóide.

As regras da competição *RoboCup NAO* são de livre consulta em (8).

1.3 Motivação e Objectivos

A constante evolução da tecnologia sugere que cada vez mais utilidades podem ser dadas aos robôs. Desde a industria à educação, ou apenas para fins lúdicos, são várias as finalidades com que os robôs são projectados. Contudo, o conhecimento científico actual pouco se reflecte nas questões realmente importantes, como a segurança e a saúde do ser humano.

O trabalho realizado ao longo deste ano lectivo, no âmbito desta Unidade Curricular, surge da necessidade de estudar o robô humanóide NAO, adquirido recentemente pelo INESC[2]-Porto, e de implementar algoritmos de inteligência artificial autónoma e controlada, com vista à participação no campeonato mundial de futebol robótico *RoboCup NAO 2011*, em Istambul – uma prova que eleva a fasquia do futebol robótico jogado na FEUP[3], não só por introduzir fortes restrições no domínio da localização, como também pelo facto de pela primeira vez serem utilizados robôs humanóides, onde o domínio do equilíbrio representa um desafio adicional. É necessário, portanto, a cooperação de não só outros alunos (mestrandos e doutorandos) como também docentes que, pela riqueza e diversidade de conhecimentos, tornarão certamente possível alcançar este objectivo. Por outro lado, este projecto pretende ainda corroborar os ideais de Asimov, através da criação de uma aplicação baseada na interacção entre humanos e robôs. Assim, será procurado estudar os mecanismos de teleoperação actualmente existentes, e criar uma aplicação inovadora, partindo de tecnologias não-invasivas existentes, de baixo custo, mas oferendo um controlo de elevada liberdade de manipulação.

São, por isso, objectivos desta dissertação:

- Conhecer as características morfológicas do robô NAO
- Estudar vários algoritmos de locomoção utilizados actualmente no robô NAO
- Estabelecer uma análise comparativa entre dois ou mais algoritmos de locomoção
- Implementar um método de extracção de ângulos baseado em análise numérica, passível de ser aplicado na locomoção do robô NAO
- Estudar a integração de outros sistemas em robôs humanóides, e sugerir uma aplicação inovadora de interacção entre robôs e humanos;
- Criar uma aplicação de ensino por demonstração.

1.4 Estrutura do Documento

Este documento está organizado em secções distintas, cujo conteúdo reflecte todo o trabalho desenvolvido ao longo desta Unidade Curricular. No restante da corrente secção encontra-se uma descrição sucinta de cada um desses capítulos. Neste capítulo em particular, é possível encontrar,

[2]Instituto de Engenharia de Sistemas e Computadores
[3]Faculdade de Engenharia da Universidade do Porto

além da presente secção informativa da estrutura do documento, uma referência às motivações e objectivos que sustentaram a realização desta dissertação. É também feita uma referência histórica sobre a robótica, o robô NAO, e a liga *RoboCup NAO*.

O capítulo 2 apresenta uma revisão bibliográfica, focando-se essencialmente nos princípios de locomoção humanóide, e nos métodos de controlo actualmente mais utilizados. O capítulo 3 consiste numa revisão de mecanismos de extracção de informação baseada em visão artificial, e outros meios de sensorização, com especial ênfase nas técnicas de detecção e localização de objectos. Será ainda possível encontrar uma referência ao princípio de funcionamento do Kinect, um controlador criado pela Microsoft para operar em conjunto com a consola XBox 360.

No capítulo 4 encontra-se descrita a cinemática do robô NAO – directa e inversa – sendo também feita uma análise de singularidades relevantes no contexto desta dissertação. Este capítulo termina com a descrição de um algoritmo iterativo baseado em análise numérica, que visa determinar uma combinação de actuações para as várias articulações de um manipulador, independentemente da sua configuração.

Todos os aspectos relativos à aplicação desenvolvida são abordados no capítulo 5. São descritos os requisitos identificados para o sistema, e é descrita a solução proposta para os atingir.

No capítulo 6 são descritos os vários testes realizados sobre o sistema, com vista a validar a solução proposta.

O capítulo 7 debruça-se sobre as principais conclusões alcançadas na realização do projecto, sendo também indicados trabalhos futuros que possam dar continuidade ao trabalho desenvolvido.

Capítulo 2

Locomoção Humanóide

Na robótica, um *humanóide* é descrito como um robô cuja construção e movimentos se aproximam às características de um ser humano. Neste sentido, *locomoção humanóide* constitui um meio de locomoção baseado na locomoção humana, isto é, recorrendo a dois apoios. Esta é tipicamente omni-direccional, e a sua velocidade é variável. O movimento humanóide pode ainda ser descrito como um movimento em três planos:

- **Sagital:** Define os movimentos *"andar para a frente"* e *"andar para a trás"*;
- **Frontal**[1]**:** Define o movimento *"andar de lado"*;
- **Transversal:** Define o movimento *"rodar sobre si"*.

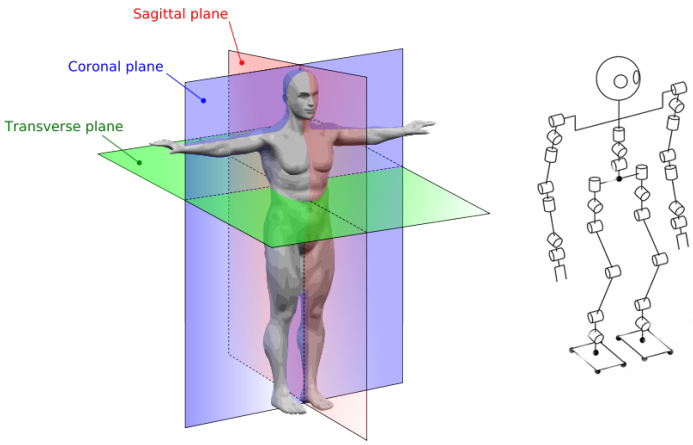

Figura 2.1: Os três planos de movimento do corpo humano *(esquerda)* e vista esquemática *(direita)*

A locomoção bípede é vantajosa, comparativamente à locomoção sobre rodas, não só pela maior agilidade dos robôs humanóides em evitar obstáculos, como pela capacidade de poderem facilmente serem empregados como substitutos nas tarefas do ser humano. Por outro lado, esta

[1] ou Coronal

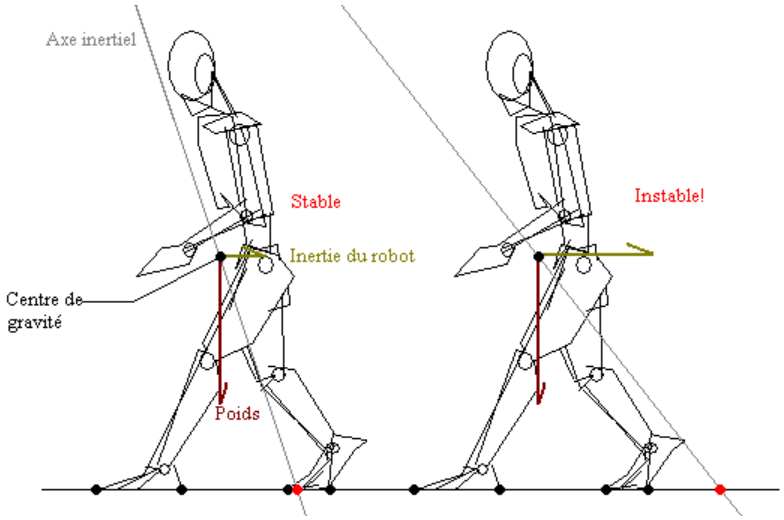

Figura 2.2: Conceito do ZMP (11)

implica a dificuldade acrescida de estudar uma dinâmica do robô significativamente mais complexa. São vários os métodos existentes que abordam esta questão, e que propõem uma solução para ela. Estes podem contudo ser classificados em dois grupos:

- **Métodos baseados em modelos:** Consistem na utilização de um modelo físico do robô, e do projecto de um controlo da locomoção baseado nesse modelo.
- **Métodos não baseados em modelos:** Baseiam-se na utilização de informação proveniente de sensores para tomar decisões em relação à trajectória a executar.

Em (9), Nima Shaffi apresentou alguns dos métodos mais utilizados e mais referenciados na literatura corrente, e que constituem o restante desta secção.

2.1 Zero Moment Point

O *Zero Moment Point* (ZMP) foi apresentado em 1968 por Miomir Vukobratović, em Moscovo, e consiste num método que permite determinar se uma dada combinação das articulações de um robô gera (ou não) um movimento estável. Por outro lado, este algoritmo é ainda imune a factores externos, tais como a inclinação do solo, ou ainda outras forças que sejam aplicadas ao robô, nomeadamente quando este é empurrado, ou quando transporta objectos (10).

Por definição, o ZMP permite encontrar o ponto no apoio do robô com o solo cujo momento produzido é nulo. Este é, portanto, o ponto onde a soma de todas as forças inerciais se anulam.

Alguns investigadores nesta área assumem que o desafio da locomoção bípede pode ser representado como o equilíbrio de um pêndulo invertido (12). Desta forma, se um corpo esta suportado por ambas as pernas, e não está parado, a assunção de que se trata de um pêndulo invertido em desequilíbrio implica também que exista uma trajectória que o possa corrigir, cujo algoritmo ZMP permite calcular – a trajectória que permitirá ao robô recuperar a estabilidade.

O passo final deste algoritmo consiste na análise da cinemática inversa, que permite obter uma combinação de actuações ao nível das articulações que satisfaça esta trajectória.

As coordenadas do ZMP consistem num par (P_x, P_y), cartesiano, representado no solo. Estas são dadas pelas expressões 2.1 e 2.2.

$$P_x = \frac{\sum_{i=1}^{n} m_i(\ddot{z}+g)x_i - \sum_{i=1}^{n} m_i\ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\ddot{\Omega}_{iy}}{\sum_{i=1}^{n}(\ddot{z}+g)m_i} \tag{2.1}$$

$$P_y = \frac{\sum_{i=1}^{n} m_i(\ddot{z}+g)y_i - \sum_{i=1}^{n} m_i\ddot{y}_i z_i - \sum_{i=1}^{n} I_{ix}\ddot{\Omega}_{ix}}{\sum_{i=1}^{n}(\ddot{z}+g)m_i} \tag{2.2}$$

Em ambas as equações, x_i, y_i e z_i representam o centro de massa da ligação i, e m_i corresponde à massa de cada uma. I_{ik} representa o momento de inércia no eixo k, da ligação i, enquanto que Ω_{ik} consiste no seu deslocamento angular, no mesmo eixo k (em radianos). g é a aceleração gravítica.

De acordo com a figura 2.2, é possível concluir que qualquer ZMP P_k terá que se situar no interior do pé de apoio, para que a estabilidade seja conservada durante o movimento.

2.1.1 Marcha Omni-Direccional do Robô NAO

Não sendo uma área suficientemente estudada, a robótica humanóide é ainda um assunto em constante evolução, contendo um número alargado de desafios. Um dos mais desafiantes, contudo, reside na capacidade de conceber algoritmos de locomoção cada vez mais eficientes, mantendo a estabilidade. Assim aconteceu com o robô NAO que, na altura da comercialização do primeiro protótipo, oferecia um conjunto de funções de locomoção bastante limitado, quer na velocidade da marcha, quer na liberdade de movimentos que executava, e que a actividade de investigação e desenvolvimento permitiu aperfeiçoar.

Actualmente o NAO integra já por defeito um mecanismo de locomoção omni-direccional através de um controlador em malha fechada (13), baseado no mecanismo introduzido por S. Kajita *et al.* (14), que introduz a utilização de um controlador preditivo e a modelização da trajectória de um robô bípede como aquela que equilibra um pêndulo invertido tridimensional. Este modelo permite ao robô não só deslocar-se para a frente ao mesmo tempo que roda, como também calcular de forma dinâmica de que modo deverá caminhar de modo a não perder o equilíbrio perante cenários diferentes, tal como a existência de pequenos obstáculos no caminho.

O mecanismo de marcha proposto em (14) e implementado pela Aldebaran Robotics (13) permite ainda que cada passo da marcha não dependa de todo o histórico precedente de passos já executados, mas sim exclusivamente de um conjunto de seis etapas distintas, descritas nas secções seguintes.

2.1.1.1 Cálculo da próxima posição do pé viajante

À semelhança do que se encontra explicado na secção respeitante ao ZMP, garantir que a projecção do centro de massa no plano do chão se encontra dentro do polígono descrito pelos pés

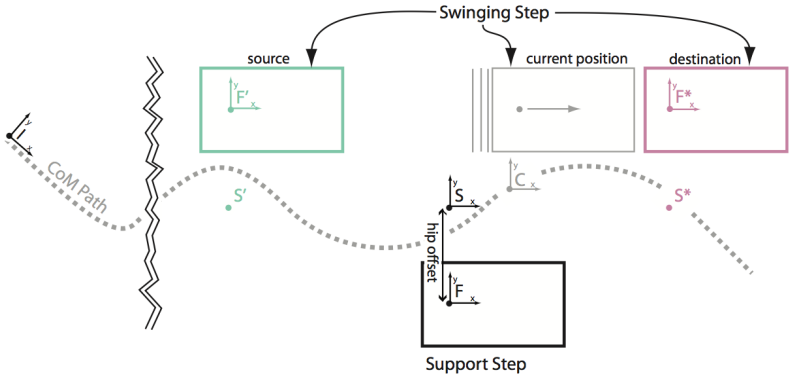

Figura 2.3: Execução de um passo (15)

garante igualmente uma marcha estável. Assim, um passo inicia e termina de forma a que o robô seja suportado por ambos os pés, e na sua trajectória um dos pés permanece no chão, suportando o peso do corpo, enquanto que o segundo pé – pé viajante – se coloca na posição final desejada (figura 2.3). A determinação das coordenadas e orientação finais do pé viajante é feita a partir de três parâmetros (x, y e θ), números escalares entre -1.0 e +1.0, que são nada mais que coeficientes dos valores máximos que estas três grandezas podem tomar.

2.1.1.2 Trajectória dos pés

Mais do que garantir que o robô não cai no início e fim de um passo, é igualmente importante preservar o equilíbrio durante a realização da trajectória que une os dois pontos. Assim, a trajectória do pé viajante é calculada por intermédio de uma interpolação cúbica com base nos pontos inicial e final, e ainda o ponto intermédio dos dois, onde o pé atinge a altura máxima (figura 2.4).

O resultado final do processo de interpolação consiste numa equação polinomial de terceira ordem, dependente de uma variável $t \in [0.0, 1.0]$, expresso pela equação 2.3.

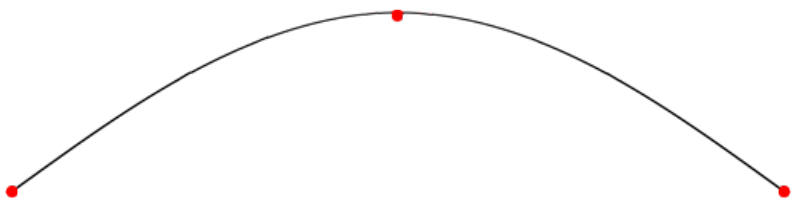

Figura 2.4: Interpolação cúbica para obtenção da trajectória do pé (16)

$$q_m(t(x)) = (1-t_m(x))y_1 + t_m(x)y_2 + t_m(x)(1-t_m(x))(a_m(1-t_m(x)) + b_m t_m(x)) \tag{2.3}$$

$$t_m(x) = \frac{x - x_m}{x_{m+1} - x_m} \tag{2.4}$$

$$a_m = k_m(x_{m+1} - x_m) - (y_m - y_m) \tag{2.5}$$

$$b_m = -k_{m+1}(x_{m+1} - x_m) + (y_{m+1} - y_m) \tag{2.6}$$

Na interpolação de três pontos, o cálculo passa pela geração de dois polinómios distintos, respeitantes a cada uma das duas ligações, e pela determinação das constantes k_0, k_1 e k_2, dados pela resolução da equação matricial 2.7.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} n_0 \\ n_1 \\ n_2 \end{bmatrix} \tag{2.7}$$

$$a_{ij} = H = \begin{cases} 0 & se\ |j-i| \geq 2 \\ \frac{1}{x_j - x_i} & se\ |j-i| < 2\ e\ |j-i| > 0 \\ 2\sum_{k=0:2;k\neq i} a_{ik} & se \| j-i| = 0 \end{cases} \tag{2.8}$$

$$n_i = 3\,\frac{y_i - y_{i-1}}{(x_i - x_{i-1})^2} + \frac{y_{i+1} - y_i}{(x_{i+1} - x_i)^2} \tag{2.9}$$

Conhecidas as $N-1$ equações, resultantes da interpolação dos N pontos, diferentes valores de x geram valores $q(t(x))$ diferentes, referentes aos valores da cota do pé ao longo do processo de interpolação. Ainda, a interpolação cúbica implementada no robô NAO gera uma trajectória suave, e livre de acelerações elevadas. Por outro lado, e uma vez que a aceleração de um motor está directamente associada ao binário que produz, a minimização desta grandeza permite tambem prolongar a longevidade dos servo-motores associados a cada uma das articulações.

A literatura contempla ainda outros métodos de geração de trajectórias, considerando não só a minimização da aceleração, mas também da velocidade e do consumo energético das articulações (17). Este último é de particular interesse, uma vez que permite que a redução do consumo de um robô permite aumentar a sua autonomia, considerando que os robôs autónomos são tipicamente alimentados por baterias. Em suma, o método apresentado consiste na descrição de uma trajectória entre dois instantes de tempo t_1 e t_2, de forma a que num terceiro instante de tempo $t_m \in [t_1, t_2]$ a aceleração é alterada, de forma a minimizar o integral da potência.

2.1.1.3 *Center of Pressure*

Enquanto que o ZMP consiste no ponto onde as forças exercidas pelo robô se anulam, o CoP[2] consiste no ponto médio da superfície que sustenta o peso do robô. Assim, quando um robô se

[2] *Center of Pressure* (em Português *Centro de Pressão*)

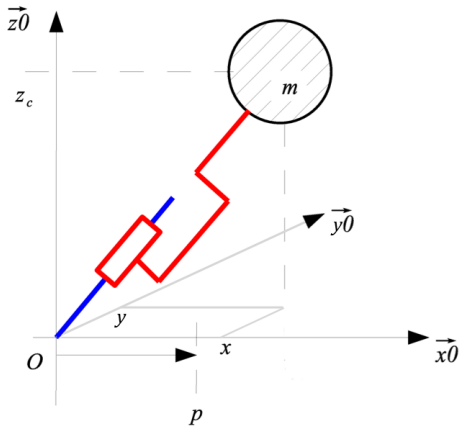

Figura 2.5: Modelização do equilíbrio de um robô como o de um pêndulo invertido linear e tridimensional (13)

encontra em equilíbrio, estes dois conceitos traduzem-se num mesmo ponto. Gouaillier explica que, no algoritmo de locomoção actualmente implementado, o CoP corresponde ao centro do pé que, durante a transladação do outro, se encontra fixo, e suporta o robô. Assim que o passo é terminado, o CoP migra para o centro do outro pé.

2.1.1.4 Pêndulo Invertido, Linear e Tridimensional

Kajita *et al.* (14) procurou explicar que o desafio de manter um robô bípede em equilíbrio pode ser modelizado como se de um pêndulo invertido se tratasse. Aplicar ao modelo a restrição adicional de que o centro de massa de um pêndulo deverá percorrer sempre uma região definida por um plano conhecido permite ainda que a dinâmica deste possa ser expressa através de um conjunto de equações que tornam o sistema linear (3D-LIPM[3] – figura 2.5).

Restringindo o movimento ao plano horizontal ($h = z_c$), as equações 2.10 e 2.11 reflectem a dinâmica do LIPM. g representa a aceleração gravítica, e m a massa do corpo. τ_x e τ_y representam os binários segundo os respectivos eixos. As componentes do ZMP (e também do CoP) – p na figura 2.5 – segundo os eixos e_x e e_y podem ser calculados como sendo o quociente entre binário aplicado e o peso que o pêndulo exerce (equações 2.12 e 2.13).

$$\ddot{x} = \frac{g}{z_c}x - \frac{\tau_y}{mz_c} \tag{2.10}$$

$$\ddot{y} = \frac{g}{z_c}y - \frac{\tau_x}{mz_c} \tag{2.11}$$

$$p_x = -\frac{\tau_y}{mg} = x - \frac{z_c}{g}\ddot{x} \tag{2.12}$$

$$p_y = -\frac{\tau_x}{mg} = y - \frac{z_c}{g}\ddot{y} \tag{2.13}$$

[3] *Three-Dimensional Linear Inverted Pendulum Mode*

O resultado que advém das operações associadas à modelização demonstrada consiste unicamente neste par (p_x, p_y). O seu conhecimento, quando comparado com a área de suporte do robô, permite determinar se a colocação do centro de massa do robô naquela posição gerará ou não uma passada estável. A geração de um padrão de marcha segue, contudo, o processo inverso: partindo de um ponto ZMP arbitrário que se sabe ser estável, determinar que combinação de forças o coloca nesse ponto.

2.1.1.5 Controlador Preditivo

O processo de marcha, conforme descrito ao longo desta secção, passa por colocar o centro de massa numa posição arbitrária, sabendo que a sua projecção no plano do chão deverá encontrar-se dentro da área que suporta o robô, durante todo o movimento. A referência do CoP sofre então alterações na sua localização no fim de cada passada, implicando que o centro de massa seja capaz de se mover em consonância com valores actualizados, e também valores futuros do CoP.

O controlador preditivo surge dessa mesma necessidade de utilizar estimativas de valores futuros do CoP no cálculo da referência actual desejada do centro de massa do robô. Tal cálculo é feito partindo das leis da cinemática de corpos (equação 2.14 e respectivas derivadas), armazenando-as sob a forma de uma matriz que reflicta a evolução discreta do centro de massa do corpo (2.15), e do CoP (2.16). $x_k = [x_k(T)\dot{x_k}(T)\ddot{x_k}(T)]^T$, e T é o período de amostragem. Kajita considerou a entrada do sistema como sendo a derivada da aceleração em relação ao tempo ($u_x = \dddot{x_k}$).

$$x(t) = x_0 + v_0.t + \frac{a.t^2}{2} \tag{2.14}$$

$$x_{k+1} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix} \dddot{x_k} \tag{2.15}$$

$$p_k = \begin{bmatrix} 1 & 0 & \frac{z_c}{g} \end{bmatrix} x_k \tag{2.16}$$

A combinação das equações 2.15 e 2.16 permite obter uma expressão para determinar os valores seguintes de p.

$$\begin{aligned} p_{k+1} &= \begin{bmatrix} 1 & 0 & \frac{z_c}{g} \end{bmatrix} x_{k+1} \\ &= \begin{bmatrix} 1 & T & \frac{T^2}{2} - \frac{z_c}{g} \end{bmatrix} x_k + \begin{bmatrix} \frac{T^3}{6} - \frac{Tz_c}{g} \end{bmatrix} \dddot{x_k} \end{aligned} \tag{2.17}$$

A solução do previsor passa pela determinação da combinação dos N estados do CoP que minimiza a aceleração do centro de massa, desde que todos se mantenham dentro do polígono que determina a sua trajectória como estável.

2.1.1.6 Cinemática Inversa

Conhecidas as trajectórias de ambos os pés e do centro de massa (tronco), o processo de análise de cinemática inversa permite conhecer, ao longo de toda a execução de uma passada, os valores a aplicar às varias articulações, tanto do pé de suporte como do pé viajante. Este estudo encontra-se descrito de forma exaustiva no capítulo 4.

2.2 Central Pattern Generator

Este método de modelo livre de geração de marcha baseia-se no princípio de que os movimentos da locomoção humanóide podem ser gerados a partir de sinais de controlo que aproximam o mais possível a trajectória de um robô àquela que é descrita por um ser humano. Num nível de abstracção mais elevado, Inoue *et al*. (18) referiu, por exemplo, que o movimento de uma cobra poderia ser aproximado por uma sinusóide, e que este efeito poderia ser alcançado por aplicar sinais sinusoidais desfasados às varias articulações.

O *Central Pattern Generator* (CPG), por se reduzir a uma manipulação das articulações em função do movimento que se pretende descrever, é vantajoso na medida em que não implica a utilização de quaisquer sensores. Todavia, está francamente limitado, pois da mesma maneira que o ser humano descreve um determinado movimento quando anda em linha recta num espaço plano, o mesmo não acontece quando desce uma escada, ou quando corre. Desta forma, o CPG é adequado e eficaz apenas no âmbito do movimento para o qual foi projectado, e num cenário para o qual foi idealizado.

No entanto, investigadores nesta área sugeriram melhorias significativas, através da alteração da morfologia deste método para uma baseada em modelos de controlo adaptativos, que pudessem utilizar informação sensorial para corrigir a trajectória, adequando-a às necessidades específicas do cenário em questão. Hopf, Matsuoka ou Ekeberg são, a título exemplificativo, três modelos matemáticos associados a redes neuronais, utilizados para gerar padrões de marcha através de métodos adaptativos. Estes partilham o princípio comum de que a cada articulação é atribuído um controlador – um conjunto de neurónios oscilatórios.

2.2.1 Oscilatório de Hopk

Hopk é um dos modelos matemáticos que visa caracterizar cada neurónio i, associado a uma articulação i, segundo as duas equações diferenciais 2.18 e 2.19.

$$\dot{x_i} = (\mu_i - r_i^2)x_i - \omega y_i \tag{2.18}$$

$$\dot{y_i} = (\mu_i - r_i^2)y_i - \omega x_i \tag{2.19}$$

Nestas equações, $r = \sqrt{x^2 + y^2}$. ω determina a frequência angular do oscilador, e μ é um coeficiente de amplitude de resposta. ω, μ_i são portanto parâmetros importantes na definição da

estabilidade dos sinais de locomoção gerados pela rede neuronal, assim como os estados iniciais de x_i e y_i.

No caso da locomoção humanóide, a figura 2.1 relembra que cada perna contém um total de 5 articulações, perfazendo um total de 10 graus de liberdade, e 20 equações oscilatórias. Este número não só aumenta o esforço computacional no cálculo das várias equações diferenciais, como implica um acréscimo de parâmetros no sistema. A equação 2.20 representa uma re-edição da equação 2.18, acrescentando um novo parâmetro ε_i. Este representa o coeficiente de acoplamento entre duas articulações i e $i-1$ (válido para movimentos segundo o eixo x), permitindo expressar trajectórias numa articulação de uma perna em função da trajectória da articulação homóloga na outra perna, e reduzindo o não só o número de parâmetros a introduzir ao sistema, como o número de equações diferenciais a calcular.

$$\dot{x_i} = (\mu_i - r_i^2)x_i - \omega y_i + \varepsilon_i x_{i-1} \tag{2.20}$$

Os modelos baseados no CPG são matematicamente robustos, e procuram tirar partidos do método da aprendizagem que define uma rede neuronal. São por isso bastante adequados quando é procurada uma fácil adaptação a novos cenários de operação (19). Contudo, embora o esforço computacional do processo de aprendizagem de uma rede neuronal seja irrelevante, as equações diferenciais propostas por este modelo já não o são, tornando difícil o processamento em cada sinapse da rede. Por outro lado, a relação de não linearidade entre os parâmetros de entrada deste sistema e a saída esperada torna árduo o trabalho de os afinar de forma a optimizar o resultado esperado.

2.3 Truncated Fourier Series

Tal como o método CPG, existem outras alternativas baseadas em funções matemáticas, que permitem descrever a locomoção de um corpo através da sua modelização em expressões matemáticas. Em concreto, a aplicação das séries de Fourier surge da constatação de que um corpo humanóide em marcha realiza um movimento periódico (figura 2.6) (9). Sabe-se que sinais de natureza periódica podem ser decompostos numa combinação de sinais sinusoidais através da expansão de Fourier, pela equação 2.21 (20).

$$F(t) = \frac{a_0}{2} + \sum_{i=1} (a_i cos(i\omega t) + b_i sin(i\omega t)) \tag{2.21}$$

Na equação acima, a_0 representa a componente contínua do sinal. ω representa a frequência angular mais baixa de todas as que compõem o sinal. Por definição, a equação 2.21 é aplicada considerando um somatório de um número infinito de componentes sinusoidais. O cálculo de um somatório de parcelas finitas reduz o esforço computacional, apesar de acarretar com isso

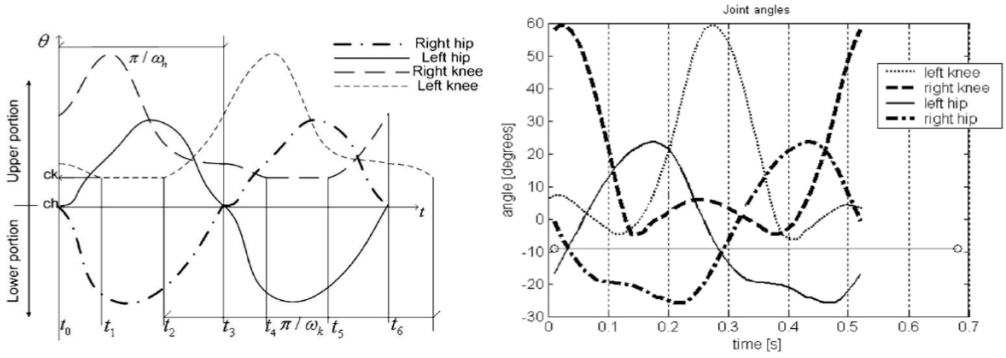

Figura 2.6: Aproximação da trajectória (linear e angular) das articulações da bacia e joelhos por funções periódicas (9)

uma perda de rigor de resultado que é imprescindível para modelizar a locomoção humanóide. Portanto, uma versão alterada deste princípio é utilizada – *Truncated Fourier Series*.

2.3.1 Geração de Trajectórias

A figura 2.6 sugere que os sinais que caracterizam o movimento das articulações podem ser divididos em duas componentes – *superior* e *inferior*, separadas pelo valor C_h, que representa o *offset* (altura inicial das articulações da bacia), da mesma maneira que C_k representa o *offset* (altura) das articulações dos joelhos. Note-se também que os movimentos dos lados direito e esquerdo estão relacionados por uma diferença de fase equivalente a metade do período, sugerindo que é possível conhecer a trajectória a aplicar às articulações de um membro apenas pelo conhecimento das trajectórias homólogas do outro membro. As equações 2.22 até 2.25 permitem conhecer as trajectórias a aplicar a cada articulação do joelho (θ_k) e da bacia (θ_h), de forma ao corpo conseguir descrever um movimento predefinido. Nestas equações, $+$ e $-$ representam respectivamente as porções superior e inferior dos membros, de acordo com a figura 2.6.

$$\theta_k^+ = C_k + C_i \sum_{i=1}^{n} sin(i\omega_k t_2), \omega_k = \frac{2.\pi}{\Gamma_k} \tag{2.22}$$

$$\theta_k^- = C_k \tag{2.23}$$

$$\theta_h^+ = C_h + A_i \sum_{i=1}^{n} sin(i\omega_h t_6), \omega_h = \frac{2.\pi}{\Gamma_h} = \omega_k \tag{2.24}$$

$$\theta_h^- = C_h + B_i \sum_{i=1}^{n} sin(i\omega_h t_6), \omega_h = \frac{2.\pi}{\Gamma_h} = \omega_k \tag{2.25}$$

A definição de uma trajectória é portanto uma função dependente de três coeficientes (A_i, B_i e C_i), do *offset* das articulações (C_k e C_h), e do conhecimento dos instantes t_2 e t_6 que representam, respectivamente, os instantes em que os joelhos esquerdo e direito finalizam a função de perna de suporte. Quaisquer outros instantes t_k não são necessários, uma vez que podem ser determinados pelo cruzamento dos quatro sinais com os valores de referência C_k e C_h. Em suma, quaisquer optimizações na marcha do robô dependem exclusivamente da afinação destes parâmetros.

2.3.2 Melhorias

Lang *et al.* propõe um algoritmo que combina a aproximação matemática do TFS com a robustez do ZMP – o GAOFSF[4] (21). Este algoritmo permite gerar uma marcha estável, tanto para cenários planos, como para cenários inclinados, sem necessidade de recorrer à cinemática inversa, ou à parametrização de diversas variáveis (como acontece com o CPG). Por outro lado, variando a frequência de cada um destes sinais (figura 2.6) variará, consequentemente, a velocidade do passo do robô.

Em 2009, Nima Shaffi alargou o conceito do TFS, anteriormente aplicado numa locomoção baseada somente no movimento das pernas, sugerindo que este também poderia ser aplicado ao movimento dos braços, e que a combinação de todos eles geraria uma locomoção não só mais estável, como mais elegante (22). Mais tarde, em 2010, propôs um algoritmo de geração de movimentos ao nível das articulações da bacia que permitiu produzir movimentos segundo o plano coronal (23).

[4] *Genetic Algorithm Optimized Fourier Series Formulation*

Capítulo 3

Sistema de Visão Computacional

Melhorar a autonomia de um robô é um dos principais desafios com que os investigadores na área da robótica se deparam. De facto, um robô é verdadeiramente autónomo quando verifica as seguintes condições (24):

- Obtém informações sobre o meio que o rodeia
- Opera por um período de tempo razoável sem a intervenção do ser humano
- Desloca-se ou move parte do seu corpo sem o auxílio do ser humano
- Evita situações que representem um potencial perigo para o ser humano ou para si próprio

Os robôs, quando não são equipados com dispositivos de sensorização adequados, dificilmente conseguem cumprir com sucesso as tarefas para as quais foram concebidos. Senão, pense-se na eficiência de um robô que possa andar, mas não consiga detectar uma parede; ou que possa agarrar objectos, mas não consiga determinar a sua localização exacta. Efectivamente, um sistema de sensorização robusto é crucial para que os restantes sistemas funcionem em sintonia. Recorde-se, por exemplo, os métodos de locomoção estudados no capítulo 2. O método CPG sugere a implementação de uma rede neuronal, cujas entradas consistem em leituras sensoriais. Estas são utilizadas para uma evolução do processo de aprendizagem, e a descoberta de uma nova combinação das articulações que gera uma melhor e mais estável trajectória. Por sua vez, o método TFS recorre a sensores para determinar a trajectória descrita pelas articulações. De acordo com a figura 2.6, estas leituras são necessárias para activar diferentes etapas do processo que caracteriza um ciclo da marcha do humanóide.

Enquanto que o capítulo 2 procurou documentar os progressos verificados no âmbito do terceiro ponto, com relevância para o projecto a desenvolver, o presente capítulo procurará elucidar sobre as soluções actualmente descobertas no âmbito do primeiro, focando-se particularmente na visão artificial.

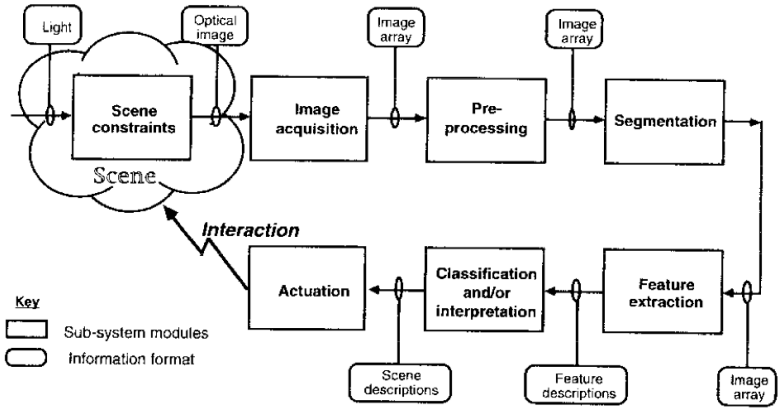

Figura 3.1: Modelo Genérico de um Sistema Baseado em Visão (26)

3.1 O Processo de Análise de Imagens

A visão é um dos sentidos humanos mais avançados e mais complexos, sendo essencial para a percepção do ambiente que nos rodeia. *Visão artificial* é o conceito que visa dotar máquinas da capacidade de emular esse sistema de percepção do ser humano, não só permitindo que estas consigam ver o Homem vê, como também fazer com que consigam interpretar essas mesmas informações. Este é um processo que compreende diversas fases (figura 3.1) (25).

3.1.1 Aquisição de Imagem

A aquisição é a primeira fase que compreende o processo de visão artificial, e consiste na captura de uma imagem (*frame*), através de uma câmara digital. A construção de uma imagem em formato digital resulta de dois processos. O primeiro – *amostragem* – consiste na discretização das coordenadas espaciais, enquanto que o segundo – *quantificação* – visa discretizar as amplitudes (figura 3.2) (26). O resultado final deste processamento é uma matriz de duas ou três dimensões (consoante se trata de uma representação a preto/branco ou a cores), em que cada célula contém o valor de cada unidade elementar da imagem (*píxel*).

Esta é uma fase de extrema importância, uma vez que a qualidade da imagem adquirida condiciona, evidentemente, a qualidade do resultado final. Como tal, é necessário ter em conta diversos factores aquando do dimensionamento de uma câmara digital, em função dos requisitos mínimos necessários para o sistema de aquisição, e do preço (27):

- Resolução (quantidade de pixéis por imagem);
- Número de bits por píxel;
- Número de *frames* captadas por segundo;
- Captação de imagens a cores ou a preto/branco;
- Representação da cor (RGB, HSI, ...).

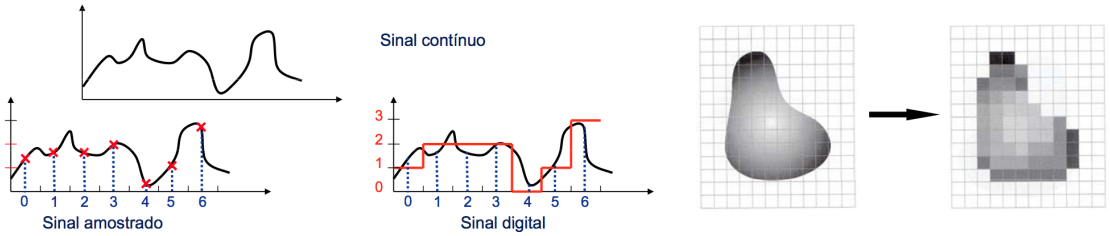

Figura 3.2: Aquisição de Imagens (26)

3.1.2 Pré-Processamento

O objectivo desta fase é o melhoramento da imagem adquirida através da eliminação de degradações e ruído indesejado, ou através do realce de características importantes. Este resultado pode ser alcançado de diversas maneiras (25). Alguns exemplos são descritos abaixo.

3.1.2.1 Transformação de intensidade

Uma transformação de intensidade consiste na passagem de cada píxel por uma função de transformação linear $f(x) = ax + b$, em que a é o coeficiente de contraste, e b é o coeficiente de brilho.

3.1.2.2 Equalização de histograma

Um histograma consiste numa representação gráfica que reflecte o registo de ocorrências de determinados níveis de cor numa imagem, e é uma característica amplamente utilizada no reconhecimento artificial de particularidades numa imagem.

Uma equalização de histograma visa diferenciar os valores mais frequentes dos píxeis que compõem uma imagem. É adequada para alargar a gama de representação e aumentar o contraste.

3.1.2.3 Limiarização

Limiarizar uma imagem consiste numa nova representação da mesma, cujos valores são alterados para um conjunto arbitrário e limitado de novos valores, através do estabelecimento de *thresholds* (figura 3.3). Dado um conjunto de valores $v_0, ..., v_n$ e os respectivos limiares $0, t_1, ..., t_n$, cada píxel com valor x será transformado num valor y tal que $y = v_k$, com $x \in [t_{k-1}, t_k]$.

3.1.2.4 Convolução Discreta

O ruído é caracterizado pela existência de um pequeno número de píxeis, geralmente isolados, cujo valor é discrepante da sua área envolvente. Embora dificilmente seja perceptível a olho nu, a sua existência em formato digital pode levar a que um robô que procure uma dada informação, a encontre por engano nestes píxeis. A maioria dos filtros recorre à convolução discreta, que procura minimizar a existência destas impurezas. Esta operação é baseada numa matriz – matriz

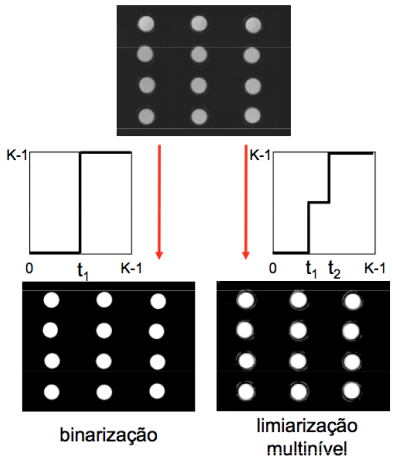

Figura 3.3: Limiarização de uma imagem (25)

de convolução (figura 3.4) – que calcula o valor de cada ponto (x,y) da nova imagem como uma combinação linear dos pontos vizinhos desse ponto (equação 3.1).

$$q(x,y) = \frac{\sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} w_{(3i+j+5)} \cdot p(x+j, y+i)}{\sum_{k=1}^{k=9} w_k} \tag{3.1}$$

A convolução discreta pode gerar resultados bastante diferentes, consoante a matriz de convolução escolhida. Matrizes baseadas em operadores de diferença, por exemplo, são utilizadas para detectar contornos. A figura 3.5 representa as transformações associadas a operadores de detecção de contornos segundo os dois eixos x e y, onde os píxeis de valores mais elevados correspondem a pontos cuja diferença com os pontos vizinhos é também ela elevada.

Outras técnicas no âmbito das operações locais podem ser aplicadas, nomeadamente a utilização de operadores de ordem, média e mediana, mínimo ou máximo, ou ainda de filtros (por exemplo filtros gaussianos).

W1	W2	W3
W4	W5	W6
W7	W8	W9

Figura 3.4: Matriz de convolução

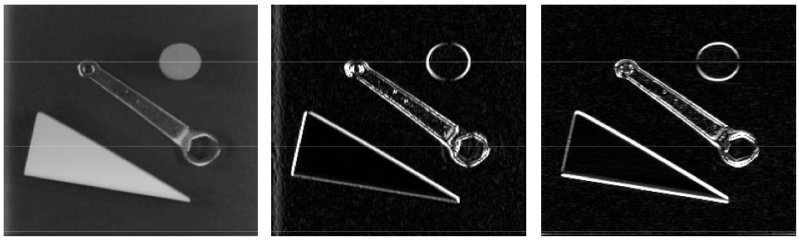

Figura 3.5: Detecção de contornos numa imagem (25)

3.1.3 Segmentação

O processo de segmentação tem como principal objectivo a divisão da imagem em blocos que permitam simplificar o processo de análise e reconhecimento de informação. Este processo é abundantemente utilizado no âmbito da localização de padrões geométricos e figuras (28). A separação da imagem em diferentes segmentos é feita com base num critério (cor, intensidade, textura, etc) que é comum a cada píxel dessa região (figura 3.6).

A segmentação de uma imagem pode ser feita de recorrendo a várias técnicas. Enumeram-se abaixo algumas, pela sua popularidade.

- **Segmentação baseada em histograma:** Limiarização com base na análise do histograma. O valor do *threshold* é determinado de forma a que o número de pontos abaixo e acima desse valor sejam iguais

- **Segmentação baseada em agrupamento:** Agregação de N regiões mutuamente exclusivas, com base na análise de N características

- **Crescimento de regiões:** Abordagem iterativa que consiste na expansão de regiões a partir de um conjunto de pontos iniciais, mediante a obediência a um conjunto de condições (critérios de agregação)

Figura 3.6: Segmentação de uma imagem

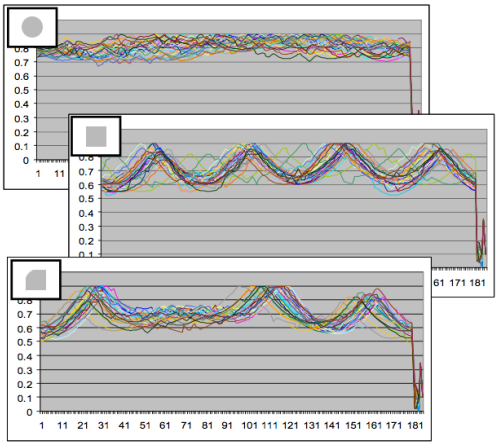

Figura 3.7: Detecção de formas geométricas baseada na distância dos pontos da orla ao centróide (29)

- **Segmentação de orlas:** Baseado na aplicação de operadores de convolução sensíveis às variações de intensidade

3.1.4 Detecção de Formas Geométricas

3.1.4.1 Distância dos pontos da orla de um objecto ao seu centróide

A detecção de formas geométricas é uma das alternativas de segmentação de uma imagem, baseada nas propriedades geométricas dos objectos no meio, e recorre a várias técnicas, baseadas no estudo dos contornos de vários objectos. Peña *et al.* (29) descreve uma dessas técnicas, baseada na comparação da distância de vários pontos da periferia de um objecto com o seu centróide. A evolução desta variável ao longo do varrimento do contorno permite determinar a forma geométrica representada. Neste estudo, no âmbito da detecção de peças sólidas numa linha de produção, foram consideradas três formas geométricas. A figura 3.7 reflecte o processamento de várias peças, com estas três formas, com tamanhos e orientações diferentes. Em (29) é ainda sugerido um aumento da robustez deste método através da implementação de um método dinâmico de classificação, recorrendo a redes neuro-difusas.

3.1.4.2 Tranformada de Hough

Por vezes as imagens podem ser capturadas com algum ruído que inviabiliza a classificação de objectos com base no método exposto em 3.1.4.1. De facto, são várias as razões que podem levar a que uma imagem não possua qualidade suficiente, e que implique não existir um número suficiente de pontos bem definidos na orla de um objecto:

- Resolução demasiado baixa

- Captura da imagem em movimento
- Iluminação deficiente
- Cores do objecto e do fundo semelhantes

A transformada de Hough visa colmatar estas falhas apresentadas nos operadores detectores de orlas, através de um algoritmo que, para um dado conjunto de pontos, procura deduzir uma equação paramétrica que passe por todos eles. O princípio deste método baseia-se na substituição da representação de um ponto (x,y) por uma curva da forma $\rho = x.cos\theta + y.sin\theta$. Ao ponto de intersecção entre as duas curvas (que representam dois pontos) corresponde aos parâmetros (ρ', θ'), em que em que ρ' é a distância da origem ao ponto mais próximo da recta, e θ' o ângulo que esta faz com o eixo de referência, no referencial $x-y$. Estes valores permitem re-escrever quaisquer rectas da parametrizadas em (ρ, θ), da forma $y = ax+b$, com $a = -tg\theta$ e $b = \frac{\rho}{sin\theta}$ (25; 30).

Este é, contudo, um método computacionalmente pesado quando o objecto em análise é composto por um número elevado de pontos. Este motivo foi suficiente para que fossem propostos outros métodos (Kimme ou O'Gorman, por exemplo), ou mesmo versões modificadas desta transformada (31).

3.2 Reconhecimento de Objectos Baseados em Visão

A visão artificial constitui de facto uma mais valia para um robô. Permite que este se localize no espaço, e permite que este localize outros agentes presentes no cenário. Por isso, um pré-processamento adequado da imagem é indispensável para garantir o sucesso das operações subsequentes.

É sabido que um dos principais problemas que motiva a investigação actual na área da robótica prende-se com a sua autonomia, e que duas das mais imediatas soluções para esta questão são naturalmente a instalação de baterias de maior capacidade, ou a redução do consumo energético do robô. Prajin Palungsuntikul e Wichian Premchaiswadi tomaram esta constatação como motivação para o desenvolvimento de uma solução que permitisse, com uma câmara e um processador de baixo custo e baixo consumo, capturar uma bola em movimento. Este processo consiste em cinco etapas (32):

- Captura da imagem digital em formato RGB[1];
- Conversão da cor de cada píxel para o formato HSI[2];
- Cálculo do histograma;
- Determinação do centro do objecto;

[1] *Red-Green-Blue*
[2] *Hue-Saturation-Intensity*

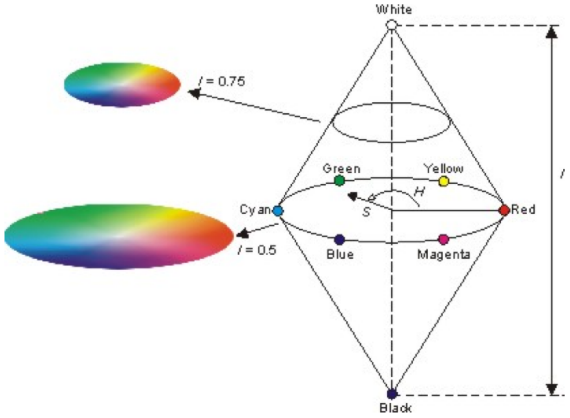

Figura 3.8: Modelo de representação de cor HSI (33)

- Cálculo da distância ao objecto.

(32) defende que o modelo de representação HSI permite que o valor da cor não dependa do valor da luminosidade, ao contrário do que acontece com a representação RGB. Desta forma, o primeiro método é adequado para a detecção de um mesmo objecto, independentemente da luminosidade do local. Assim, o cálculo da posição de um objecto cuja cor está bem definida pode ser feito analisando exclusivamente a matriz *Hue* (figura 3.8). Esta conversão é dada pelas equações 3.2 até 3.5.

$$H = \begin{cases} \theta & se\ B \leq G \\ 360 - \theta & se\ B > G \end{cases} \tag{3.2}$$

$$\theta = cos^{-1}\left(\frac{R - \frac{G+B}{2}}{[R^2 + G^2 + B^2 - (RG + GB + BR)]^{\frac{1}{2}}}\right) \tag{3.3}$$

$$S = 1 - \frac{3}{R+G+B}[min(R,G,B)] \tag{3.4}$$

$$I = \frac{R+G+B}{3} \tag{3.5}$$

O desconhecimento da cor do objecto não invalida que este consiga ser encontrado. Efectivamente, a terceira etapa enunciada por (32) sugere a análise do histograma da componente *Hue*. Esta permite determinar, entre outros valores, quais as cores que registaram uma maior ocorrência na imagem, tornando possível a determinação desta variável.

A determinação do centro é feita em duas fases. A primeira consiste em assinalar na imagem todos os píxeis cujo valor corresponde à cor desejada – a este processo dá-se o nome *segmentação*.

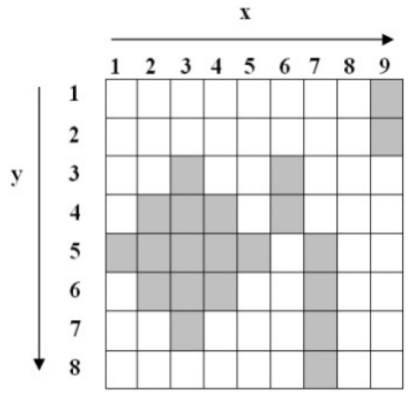

Figura 3.9: Número de píxeis de uma segmentação (32)

A segunda fase consiste em analisar todas as colunas que representam a matriz da nova imagem, e seleccionar aquela que concentre o maior número de pontos consecutivos (figura 3.9). A segunda coordenada do centro corresponde ao ponto médio entre o primeiro e último pontos.

3.3 Cálculo da Distância a um Objecto

O cálculo da distância de um objecto recorrendo exclusivamente a mecanismos de visão com uma câmara é um resultado, por si só, impossível de alcançar, a menos que o sistema disponha de mais informação do que apenas uma imagem capturada. São vários os algoritmos que a literatura apresenta para combater esta falha, nomeadamente através da utilização de múltiplas câmaras (e consequente triangulação dos seus resultados), ou métodos que resultem da combinação de múltiplos sensores (ultra-sons ou infra-vermelhos, por exemplo). De facto, a utilização de uma só câmara elimina a componente de profundidade de uma leitura, limitando-se apenas a realizar observações a duas dimensões. Um dos métodos mais tradicionalmente implementados para combater este défice de informação consiste em recorrer às dimensões do objecto em questão, e a assunção de que este se encontra pousado no chão 3.10. Este método, utilizado pelas equipas de futebol robótico *bHuman* (34) e *5DPO*, é descrito no restante desta secção.

O primeiro passo deste algoritmo consiste na segmentação da bola, e contagem dos píxeis que constituem cada coluna deste objecto (secção 3.2), sendo que ao maior destes números – $max(p)$ – corresponde o diâmetro (conhecido) da bola, em píxeis. Através deste valor é possível conhecer a resolução de cada pixel, tal que $res = \frac{2.r}{max(p)}(metro/pixel)$. Δx e Δy são agora facilmente convertidos – $\Delta_k = res.\delta_k$. Δ_k está em metros, enquanto que δ_k está em píxeis.

Este resultado corresponde à distância relativamente ao centro da imagem. A conversão deste valor para o referencial da câmara é facilmente feita pela relação trigonométrica $tang(\theta).h = l$. As equações 3.6 a 3.7 resumem o algoritmo de obtenção da posição da esfera (P_x, P_y), em relação ao robô (aplicável a qualquer forma geométrica de dimensões conhecidas).

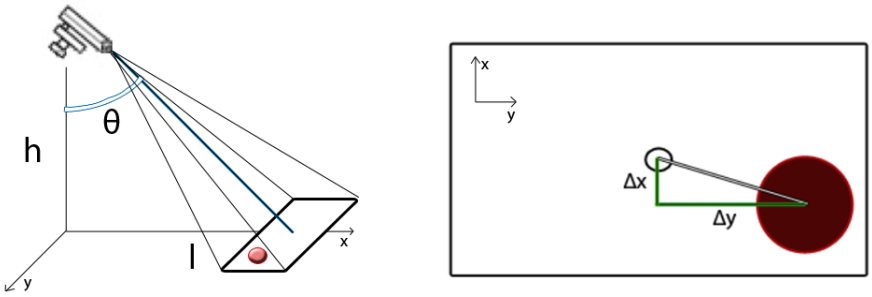

Figura 3.10: Determinação das coordenadas de um objecto, assumindo que este está pousado no chão

$$P_x = tan(\theta).h - \frac{2.r.\delta_x}{max(p)} \tag{3.6}$$

$$P_y = \frac{2.r.\delta_y}{max(p)} \tag{3.7}$$

3.4 Microsoft XBox Kinect

Kinect é o controlador lançado em Novembro de 2010 pela empresa Microsoft, disponível para a consola de video-jogos XBox 360 (35). Este dispositivo constitui um periférico que introduziu um avanço significativo na indústria do entretenimento. O controlador Kinect é constituído por duas câmaras e um laser infra-vermelhos, dispostos conforme a figura 3.11. Esta configuração permite ao controlador capturar informação visual a três dimensões. Além disso, este dispositivo possui ainda na sua base um motor, que permite articular a câmara segundo o eixo horizontal.

Desde o lançamento do Kinect, e tratando-se esta de uma plataforma fechada e patenteada para trabalhar exclusivamente com a XBox 360, a Microsoft não publicou, até à data, quaisquer

Figura 3.11: Exterior *(esquerda)* e interior *(direita)* do controlador Kinect (36)

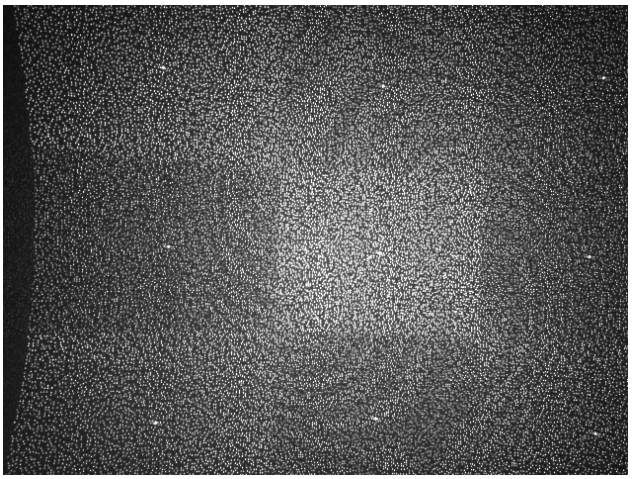

Figura 3.12: Projecção do laser infra-vermelhos, para cálculo da profundidade (38)

bibliotecas ou protocolos implementados. Por isso, todo o conhecimento actualmente disponível é fruto da investigação individual e espírito de partilha dos mais curiosos. Esta tecnologia é indubitavelmente revolucionária, sendo no entanto ainda emergente pelo que, embora se adivinhe o potencial que possa ter, não só no entretenimento como para o avanço científico que pode trazer, poucas aplicações existem actualmente. A comunidade *OpenKinect.org* (37) surge desta mesma necessidade: a de satisfazer a curiosidade de todos o que procuram saber mais sobre o Kinect, reunindo um conjunto de tutoriais e *software* que permita que qualquer computador reconheça a informação enviada pelo Kinect.

Uma vez que a XBox 360 já existe desde 2005, e o Kinect foi projectado em 2010 para ser compatível com o equipamento já existente, neste último foi incluído uma unidade de processamento de imagem, que antecede o envio da informação para a XBox. Desta forma, o controlador colecciona e analisa toda a informação visual, devolvendo o seu resultado em duas componentes: uma imagem a cores e um mapa de profundidades (36).

3.4.1 Cálculo de Profundidade

Embora a Microsoft não tenha até à data revelado quaisquer especificações técnicas sobre o modo de operação do Kinect, nem sobre como este obtém informações de profundidade, Kurt Konolige e Patrick Mihelich conduziram algumas experiências com vista à descoberta da melhor aproximação deste algoritmo (38). O restante desta secção procura abordar esse método.

O cálculo da profundidade de uma imagem recorre à câmara e ao laser infra-vermelhos. Este processo é iniciado pela projecção de uma imagem constituída por pontos pretos e brancos. Estes são distribuídos de acordo com um padrão fixo, de acordo com a figura 3.12.

A profundidade é calculada através da triangulação com um padrão conhecido. Para isso, o controlador possui uma projecção por defeito (para uma profundidade p conhecida) do projector. A profundidade de cada píxel da imagem capturada através da câmara de infra-vermelhos é obtida

fazendo um varrimento desta, através de uma janela de correlação (tipicamente de tamanho 9x9), e comparando o padrão nessa janela com o padrão conhecido. O resultado desta operação é um valor de profundidade relativamente à profundidade p – *disparidade*. A triangulação da matriz de profundidades relativas, $P_{x,y}$, com o padrão conhecido à profundidade p resulta numa matriz de profundidades em relação ao controlador.

3.4.1.1 Cálculo da disparidade

A disparidade normalizada $d(x,y)$ e a profundidade absoluta de um píxel $p(x,y)$ estão relacionadas pela fórmula $p(x,y) = \frac{b.f}{d(x,y)}$, em que b é a distância horizontal entre os sensores na câmara (aproximadamente $7.5cm$), e f é a distância focal da câmara (em píxeis). A disparidade normalizada é obtida em função da disparidade mensurada pelo kinect, $K_d(x,y)$, pela equação $d(x,y) = \frac{d_{offset} - K_d(x,y)}{8}$ (d_{offset} – de aproximadamente 1090 – é uma constante de correcção respeitante a cada dispositivo Kinect). Esta fórmula corrige o valor $d(x,y)$ utilizado na primeira equação, da qual resultaria uma profundidade infinita quando K_d fosse um valor bastante pequeno, ou mesmo nulo. O coeficiente $\frac{1}{8}$ resulta da conversão de unidades de $[pixel]$ para $[\frac{pixel}{8}]$.

Capítulo 4

Análise da Cinemática

Cinemática consiste num ramo da mecânica clássica que estuda o movimento de corpos no espaço, e é uma das bases do controlo de robôs. Na modelização de robôs, as partes que os constituem são divididas em dois grupos (39; 40): *articulações* (modelizando actuadores), aos quais correspondem 1 grau de liberdade, e *segmentos* (modelizando ligações entre duas articulações). Também as articulações podem ser classificadas em duas classes: *prismáticas* ou *rotativas*. Articulações prismáticas descrevem movimentos lineares, enquanto que articulações rotativas descrevem rotações segundo um determinado eixo (figura 4.1). Na modelização de robôs, as seguintes regras são convencionadas:

- Um robô possui n articulações e $n+1$ segmentos
- Uma articulação i une os segmentos $i-1$ e i
- Actuar na articulação i faz o segmento i mover-se
- O segmento 0 é a base, e não se move
- A cada articulação i está associado um referencial $o_i x_i y_i z_i$

4.1 Cinemática Directa

A cinemática directa é um processo que permite obter expressões matemáticas que determinam a posição e orientação das várias articulações de um manipulador, partindo do conhecimento de todas as variáveis relacionadas com as articulações – tipologia (prismática ou rotativa), comprimento dos segmentos adjacentes e valores actuados. Assim, a posição da articulação é definida por uma matriz $P^i = [x_i\ y_i\ z_i]^T$, cujos valores definem a distância nos três eixos cartesianos a partir da articulação anterior, e expresso nas coordenadas do referencial associado a essa articulação. Também a orientação é expressa em função da orientação da articulação anterior, e a sua representação consiste numa matriz de rotação R^i_{i+1}.

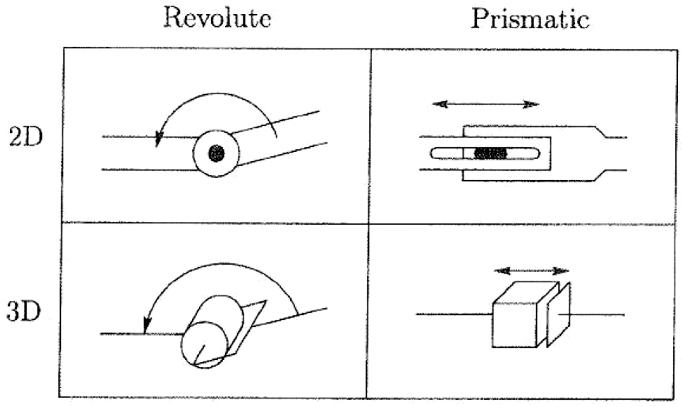

Figura 4.1: Diferentes tipos de articulações (39)

A reunião destas duas matrizes permite conhecer a matriz H_i – *Matriz de Transformação Homogénea* (equação 4.1), o que possibilita a representação de coordenadas de pontos num segundo referencial em função da sua representação num primeiro (equação 4.2). A determinação da extremidade de um manipulador (na extremidade do segmento n) em função do referencial fixo (P^n), e que define o objectivo da cinemática directa, pode assim ser determinada através da equação 4.3.

$$H_i = \begin{bmatrix} R_i & P_i \\ 000 & 1 \end{bmatrix} \tag{4.1}$$

$$P^i = H^i_{i+1} P^{i+1} \tag{4.2}$$

$$P^0 = \prod_{i=0}^{n-1} H^i_{i+1} P^n = H^0_n P^n \tag{4.3}$$

4.1.1 Método *Devanit-Hartenberg*

Devanit-Hartenberg (ou *D-H*) consiste num método analítico de modelização de um manipulador robótico, que procura determinar uma representação minimalista da posição das várias articulações. Este algoritmo, amplamente utilizado na comunidade e referenciado pela literatura, define referenciais associados a cada articulação, obedecendo a dois princípios:

1. O eixo z_n está na direcção do eixo da articulação a_n;

2. O eixo x_n é perpendicular ao plano definido pelos eixos z_n e z_{n-1};

Este método permite descrever a transformação homogénea H^i_{i+1} (ou seja, as matrizes R^i_{i+1} e P^i_{i+1}) como o produto de quatro transformações independentes (equações 4.4 a 4.7), caracterizadas respectivamente por quatro parâmetros (39):

- d**:** distância do eixo x actual ao seguinte;

- θ**:** ângulo entre o eixo x actual e o seguinte;
- r**:** distância do eixo z actual ao seguinte;
- α**:** ângulo entre o eixo z actual e o seguinte;

$$Trans(d_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.4}$$

$$Rot(\theta_n) = \left[\begin{array}{ccc|c} cos(\theta_n) & -sin(\theta_n) & 0 & 0 \\ sin(\theta_n) & cos(\theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.5}$$

$$Trans(r_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline r_n & 0 & 0 & 1 \end{array}\right] \tag{4.6}$$

$$Rot(\alpha_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_n) & -sin(\theta_n) & 0 \\ 0 & sin(\theta_n) & cos(\theta_n) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.7}$$

$$\begin{aligned} H_n^{n-1} &= Trans(d_n).Rot(\theta_n).Trans(r_n).Rot(\alpha_n) \\ &= \left[\begin{array}{ccc|c} cos(\theta_n) & -sin(\theta_n)cos(\alpha_n) & sin(\theta_n)sin(\alpha_n) & r_n cos(\theta_n) \\ sin(\theta_n) & cos(\theta_n)cos(\alpha_n) & -cos(\theta_n)sin(\alpha_n) & r_n sin(\theta_n) \\ 0 & sin(\alpha_n) & cos(\theta_n) & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned} \tag{4.8}$$

4.2 Cinemática Inversa

Enquanto que a cinemática directa permite conhecer as coordenadas das n articulações no espaço cartesiano, com base nos valores a elas aplicados, nos ângulos e comprimentos dos respectivos segmentos, o controlo de robôs é geralmente feito recorrendo à cinemática inversa. Desta forma, sabendo as coordenadas das articulações é possível calcular um conjunto de todos os valores d_p e θ_r (com $p \in P$ – conjunto das articulações prismáticas – e $r \in R$ – conjunto das articulações rotativas), tais que essas coordenadas são alcançadas.

Contudo, a observância de tais equações não implica que a solução seja única. De facto, múltiplas combinações de ângulos podem permitir ao robô atingir a mesma posição e orientação

da extremidade. A determinação da configuração correcta das articulações passa portanto pela resolução de um sistema altamente não linear, baseado nas restrições e limitações físicas do robô.

O processo de resolução de um problema tradicional de cinemática inversa parte das coordenadas dos pontos inicial e final, dos comprimentos de todos os segmentos envolvidos, e do tipo de articulações, sendo o conhecimento destas variáveis necessário e suficiente para a descoberta de pelo menos uma solução. No entanto, dois fenómenos podem conduzir à existência de infinitas soluções:

- **Redundância**, quando o número de articulações excede o número de graus de liberdade. A complexidade da cinemática inversa é tanto maior quando maior for o número de articulações. Seis é o número mínimo de articulações necessário para atingir seis graus de liberdade e, portanto, possibilitar ao robô atingir, dentro do volume de trabalho, qualquer posição e orientação do segmento da extremidade.
- **Singularidade**, quando várias combinações de uma ou mais articulações sobrepostas colocam a articulação seguinte na mesma posição. Uma singularidade pode ser identificada através da análise do Jacobiano do manipulador, e da determinação do conjunto de valores das articulações que anula o seu determinante.

4.3 Cinemática do NAO

O NAO é um robô humanóide com 5 articulações em cada braço, 2 na cabeça, 5 em cada perna, 1 na bacia e 1 em cada mão, perfazendo um total de 25 graus de liberdade[1] de manipulação. Tal como a figura 4.2 indica, robôs antropomórficos podem ser decompostos como se de 4 braços robóticos se tratassem – 2 braços e 2 pernas. De facto, este foi o conceito aplicado pela Aldebaran Robotics aquando da modelização do robô.

A análise da cinemática inversa do robô NAO permite, à semelhança do que foi referido na secção 4.2, obter expressões que permitem relacionar os valores a aplicar nos vários actuadores associados às articulações, em função das respectivas posições. Por isso, tratando-se desta plataforma específica, em todo o processamento matemático foi incluído o *offset* existente por defeito nas articulações. A figura 1.1, na página 3, descreve a posição e orientação de todas as articulações do robô, quando os valores nelas actuados são 0º.

De acordo com a figura 4.2, um braço pode ser modelizado através de 5 articulações rotativas, das quais 2 reflectem o movimento do ombro, 2 o do cotovelo, e uma última reflecte a rotação do pulso. Contudo, esta última articulação foi ignorada ao longo de todo o projecto, uma vez que a versão do robô NAO utilizada ao longo do projecto não possui qualquer actuador associado a esta.

Conforme será possível encontrar na secção 5.3.2, nesta aplicação procurou-se utilizar as bibliotecas da *OpenNI*, por ser possível retirar delas as coordenadas de referência das várias articulações, devolvidas pelo Kinect, permanentemente actualizadas. Isto permite relaxar a análise da cinemática desta aplicação, reduzindo significativamente a ocorrência de múltiplas soluções.

[1] deslocamento ou rotação independente

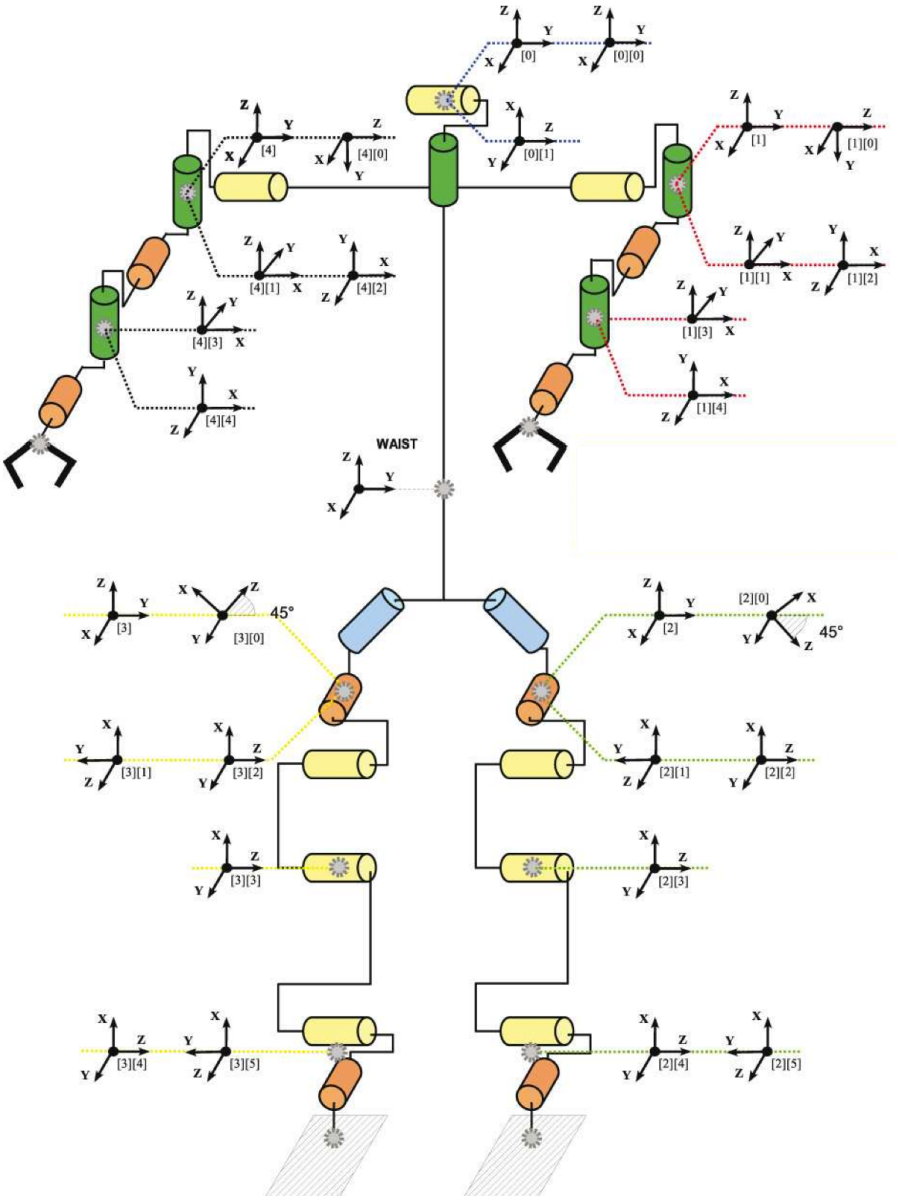

Figura 4.2: Esquemático das articulações do NAO (41)

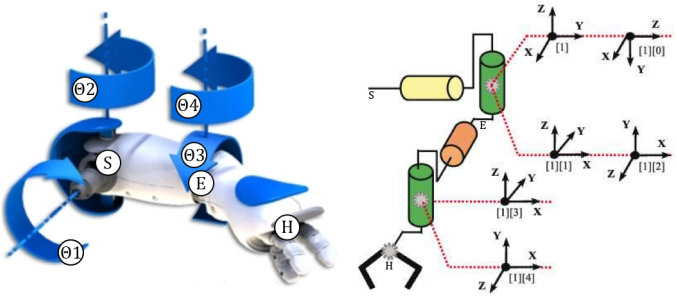

Figura 4.3: Braço do NAO, e esquemático das articulações (41)

Desta forma, todo o processo de extracção dos ângulos das articulações, tanto dos braços como das pernas, pode ser feito pela análise de cada articulação de forma independente. A determinação dos ângulos é um processo que compreende diversas fases:

- Determinação do referencial actualizado da articulação a_i da base;
- Determinação das coordenadas da articulação a_{i+1}, representadas no referencial determinado anteriormente;
- Determinação da matriz de transformação homogénea, que coloca o referencial na articulação seguinte, com a orientação desta.
- Decomposição da componente de rotação da matriz de transformação homogénea num produto de duas matrizes de rotação em torno de um eixo;
- Extracção dos dois ângulos que definem cada uma das matrizes de rotação determinadas.

Este é um algoritmo robusto para a determinação dos ângulos a aplicar às articulações dos braços. No entanto é computacionalmente pesado quando o número de articulações a calcular com base neste método aumenta (em virtude do aumento do número de equações altamente não lineares), e quando restrições rígidas de tempo existem, do qual a aplicação desenvolvida é um exemplo. Optou-se portanto por recorrer a métodos menos pesados, e que pudessem tirar partido da simplificação do problema.

4.3.1 Ombros

Os ombros correspondem às articulações imediatamente acopladas ao dorso – referencial fixo – e são modelizados por duas articulações rotativas (as duas primeiras da figura 4.3). Em alternativa à análise clássica da cinemática inversa, o algoritmo proposto recorre às leis da trigonometria, permitindo a mudança de notação de pontos descritos no referencial cartesiano para o referencial esférico, de acordo com a figura 4.4. Este sistema de coordenadas permite expressar qualquer ponto em função da distância à origem, e de dois ângulos segundo dois eixos.

Partindo apenas do conhecimento das coordenadas cartesianas do ombro (S_x, S_y, S_z) e do cotovelo (E_x, E_y, E_z), expressas em relação a um referencial fixo, as equações 4.9 e 4.10 permitem determinar os dois ângulos das articulações do ombro – θ_1 e θ_2, respectivamente. Abaixo, $\Delta_i = E_i - S_i$ para qualquer eixo i.

$$\theta_1 = atan\left(\frac{\Delta y}{\Delta z}\right) \tag{4.9}$$

$$\theta_2 = \begin{cases} atan\left(\frac{\sqrt{\Delta z^2+\Delta x^2}}{\Delta y}\right) & se\ Braco\ Direito \\ -atan\left(\frac{\sqrt{\Delta z^2+\Delta x^2}}{\Delta y}\right) & se\ Braco\ Esquerdo \end{cases} \tag{4.10}$$

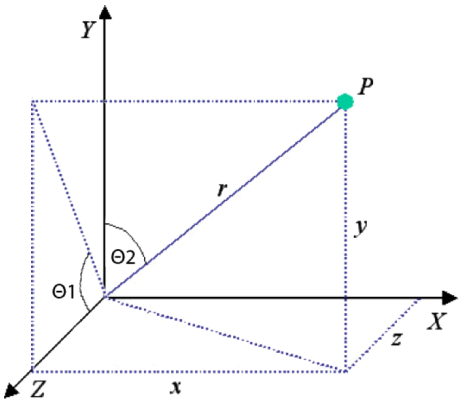

Figura 4.4: Sistema de coordenadas esféricas

4.3.2 Cotovelos

O cotovelo corresponde ao conjunto das duas articulações ligadas ao ombro através do antebraço. Dessa forma corresponde às articulações [3] e [4] da figura 4.3. Embora o cálculo do valor destas articulações siga um raciocínio semelhante ao utilizado para os ombros, este depende da orientação do referencial colocado no cotovelo, e das coordenadas da respectiva mão, representadas nesse referencial.

A orientação do referencial associado ao cotovelo varia em função dos valores aplicados às duas articulações do ombro. Portanto, a matriz que relaciona esta transformação pode ser determinada como o produto de duas rotações (figura 4.5), resultando na equação matricial 4.11, em que o eixo e_z (do cotovelo) aponta na direcção do antebraço. Este processo é obtido através de uma rotação segundo o eixo e_y de θ°, seguido de uma rotação segundo o eixo e_x de ϕ°, baseada no referencial obtido após a primeira rotação.

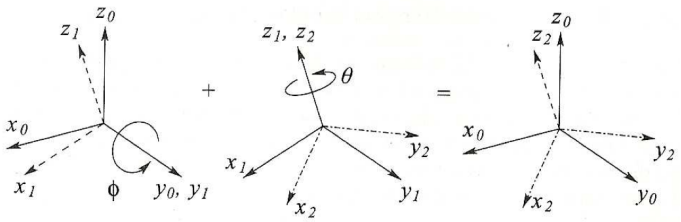

Figura 4.5: Decomposição da rotação de um referencial em dois eixos em duas rotações em um eixo, com $R_2^0 = R_1^0 R_2^1$ (39)

$$R = R_{y,\theta}R_{x,\phi} = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} = \\ = \begin{bmatrix} cos(\theta) & sin(\theta)sin(\phi) & sin(\theta)cos(\phi) \\ 0 & cos(\phi) & -sin(\phi) \\ -sin(\theta) & cos(\theta)sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \tag{4.11}$$

Uma vez que a expressão de um ponto num referencial A pode ser descrito a partir de coordenadas no referencial B através da equação $p^A = R_B^A.p^B$, e que R_B^A consiste na matriz de rotação que transforma o referencial A em B, então $p^{elbow} = R^{-1}.p^{shoulder}$, em que R^{-1} é a matriz que coloca o referencial do cotovelo no ombro (equação 4.12).

$$R^{-1} = \frac{adj(R)}{det(R)} = R^T = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ sin(\theta)sin(\phi) & cos(\phi) & cos(\theta)sin(\phi) \\ sin(\theta)cos(\phi) & -sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \tag{4.12}$$

Conhecida a matriz que permite transformar coordenadas do referencial fixo para o do cotovelo, é possível calcular os ângulos que deverão ser aplicados nas duas articulações do cotovelo, e que transportam a mão para a posição (h_x, h_y, h_z) desejada, expressa no referencial fixo. Uma vez que o primeiro ângulo – θ_3 – remete para uma rotação no eixo colinear com a orientação do antebraço, o seu valor pode ser determinado com base na projecção da mão (representada no referencial do cotovelo – ponto N obtido através da equação 4.13) no plano $o_e x_e z_e$, e a sua componente em e_y, resultando na equação 4.14.

$$\begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = R^{-1} \begin{bmatrix} H_x - E_x \\ H_y - E_y \\ H_z - E_z \end{bmatrix} \tag{4.13}$$

$$\theta_3 = atan2\left(N_y, \sqrt{N_x^2 + N_z^2}\right) \tag{4.14}$$

Para o cálculo do segundo ângulo do cotovelo – θ_4 – considerou-se a observância da desigualdade de Cauchy-Schwarz[2], e a consequência desta desigualdade na definição geométrica de um ângulo entre dois vectores (equação 4.15).

$$\theta_{A,B} = acos\frac{A.B}{||A||.||B||} \tag{4.15}$$

Uma vez que existe informação suficiente para construir dois vectores (Ombro-Cotovelo e Cotovelo-Mão), e que estes formam um plano, é possível calcular o ângulo que fazem através da equação 4.16. Note-se que, apesar da recorrência à função *acos* na determinação do ângulo θ_4

[2]Para dois vectores A e B, $A.B \leq ||A||.||B||$

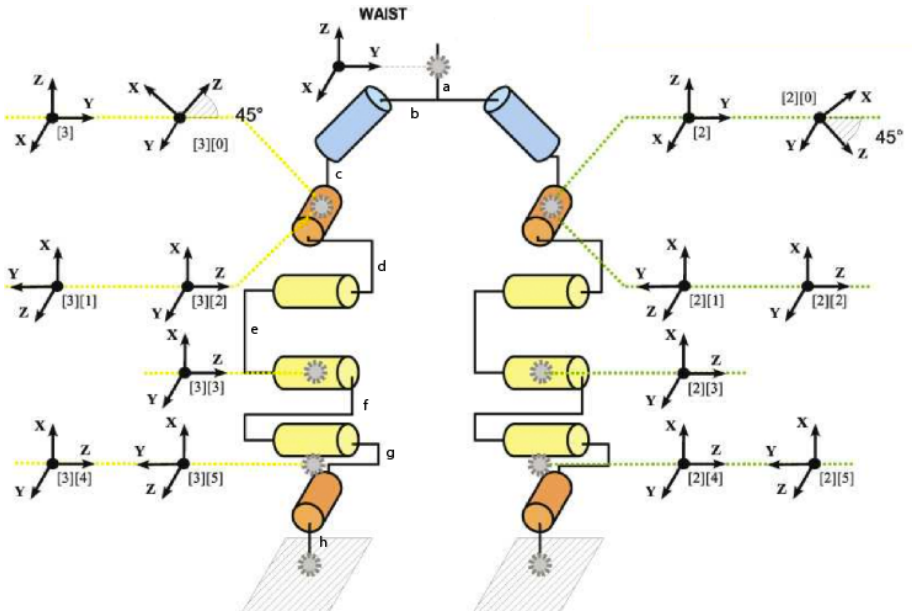

Figura 4.6: Esquemático das articulações das pernas do NAO (41)

originar duas soluções, a sua utilização no contexto deste projecto implica a existência de apenas uma solução, dado que o movimento desta articulação em particular se restringe dos 0º aos 180º.

$$\begin{aligned} A &= (E_x - S_x, E_y - S_y, E_z - S_z) \\ B &= (H_x - E_x, H_y - E_y, H_z - E_z) \\ A.B &= ||A||.||B||cos(\theta_4) \\ \theta_4 &= acos\left(\frac{A.B}{||A||.||B||}\right) \end{aligned} \tag{4.16}$$

4.3.3 Pernas

Um humanóide é constituído por duas pernas – dois conjuntos de segmentos e articulações que unem a bacia ao chão – sendo responsáveis por suportar o peso de todo e corpo, e também por gerar movimentos que permitam a deslocação deste. No caso concreto do robô NAO, as suas pernas são modelizadas com um total de 6 articulações, de acordo com a figura 4.6. Destas, as três primeiras ([0], [1] e [2]) estão directamente acopladas à bacia, enquanto que a articulação [3] define o movimento associado ao joelho, e as restantes duas emulam o movimento do calcanhar – [4] e [5].

Uma vez mais, o objectivo de estudar a análise cinemática das pernas é conseguir obter uma representação da sua extremidade – pé – e das articulações associadas em função de um referencial fixo (*waist* na figura 4.6). A secção 2.1.1 descreve de forma pormenorizada o algoritmo de locomoção que acompanha a versão mais recente do NAO: de que forma gera a trajectória de ambos

os pés durante uma passada, e qual a posição desejada, quer o centro de massa, quer do CoP, de forma a manter o equilíbro do robô durante todo o movimento.

Assim, seguindo os princípios enunciados ao longo da secção 4.1 é possível determinar uma matriz homogénea que relaciona a posição do pé em relação ao tronco (centro de massa), em função dos ângulos aplicados em cada uma das seis articulações cujos segmentos ligam, como o produto de sete matrizes homogéneas, determinadas segundo o método D-H abordado na secção 4.1.1, que reflectem as transformações resultantes das actuações nas seis articulações. As equações 4.17[3] a 4.20 reflectem respectivamente as transformações do tronco para o quadril, do quadril para o joelho, do joelho para o tornozelo e do tornozelo para o pé.

$$H_{hip}^{waist} = \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & -b \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & -a \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.17}$$

$$H_{knee}^{hip} = \left[\begin{array}{ccc|c} cos(\theta_1)\frac{\sqrt{2}}{2} & cos(\theta_1)\frac{\sqrt{2}}{2} & -sin(\theta_1) & -c.sin(\theta_1) \\ sin(\theta_1)\frac{\sqrt{2}}{2} & sin(\theta_1)\frac{\sqrt{2}}{2} & cos(\theta_1) & c.cos(\theta_1) \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} cos(\theta_2) & 0 & sin(\theta_2) & 0 \\ sin(\theta_2) & 0 & -cos(\theta_2) & 0 \\ 0 & 1 & 0 & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} cos(\theta_3) & 0 & -sin(\theta_3) & -e.sin(\theta_3) \\ sin(\theta_3) & 0 & cos(\theta_3) & e.cos(\theta_3) \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.18}$$

$$H_{ankle}^{knee} = \left[\begin{array}{ccc|c} cos(\theta_4) & -sin(\theta_4) & 0 & -f.sin(\theta_3) \\ sin(\theta_4) & cos(\theta_4) & 0 & f.cos(\theta_3) \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.19}$$

[3] válido para a perna direita

$$H_{foot}^{ankle} = \left[\begin{array}{ccc|c} cos(\theta_5) & 0 & -sin(\theta_5) & -g.sin(\theta_5) \\ sin(\theta_5) & 0 & cos(\theta_5) & g.cos(\theta_5) \\ 0 & -1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} sin(\theta_6) & 0 & -cos(\theta_6) & -h.cos(\theta_6) \\ -cos(\theta_6) & 0 & -sin(\theta_6) & -h.sin(\theta_6) \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$
$$= \left[\begin{array}{ccc|c} cos(\theta_5)sin(\theta_6) & -sin(\theta_5) & -cos(\theta_5)cos(\theta_6) & -g.sin(\theta_5) - h.cos(\theta_5)cos(\theta_6) \\ sin(\theta_5)sin(\theta_6) & cos(\theta_5) & -sin(\theta_5)cos(\theta_6) & g.cos(\theta_5) - h.sin(\theta_5)cos(\theta_6) \\ cos(\theta_6) & 0 & 0 & h.sin(\theta_6) \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.20}$$

$$H_{foot}^{waist} = H_{hip}^{waist} H_{knee}^{hip} H_{ankle}^{knee} H_{foot}^{ankle} \tag{4.21}$$

4.3.4 Singularidades

Dada a quantidade de informação existente para a análise da cinemática dos braços, para qualquer posição possível existe uma e uma só combinação das quatro articulações, à excepção de dois casos. De facto, $\theta_2 = \pm 90º$ faz com que o cotovelo atinja sempre a mesma posição, independentemente do valor aplicado em θ_1. Quando $\theta_4 = 0º$, uma singularidade surge por se estar a atingir um ponto limite do volume de trabalho.

A matriz Jacobiana consiste numa representação matricial de 6 linhas e N colunas, em que N é o número articulações de um dado manipulador. Esta matriz é composta pelas derivadas parciais das equações de cinemática, em ordem à variável a que corresponde cada articulação, e descreve a relação entre as velocidades linear e angular da extremidade e as velocidades lineares e angulares de cada uma das articulações, e que são conhecidas. Destas seis linhas, as três primeiras reflectem a relação entre as velocidades lineares nos três eixos, enquanto que as restantes reflectem a relação entre as velocidades angulares. Entre os vários aspectos em que esta matriz é útil, ela permite ainda determinar combinações de articulações que, quando aplicadas no manipulador, originam singularidades (como os dois casos acima descritos). Na matriz Jacobiana, uma singularidade é detectada quando uma combinação particular de valores reduz a característica da matriz, e estes pontos são caracterizados por geralmente neles existir uma infinidade de soluções.

A equação 4.22, abaixo determinada, consiste no Jacobiano do braço da figura 4.3. Nessa mesma figura, a representação à direita explicita a configuração obtida quando é aplicado o valor de 0º a cada articulação, sendo a base da matriz abaixo calculada. Foi também tida em consideração a representação dos referenciais associados a cada articulação segundo o método D-H. Por

outro lado, a matriz Jacobiana de manipuladores com menos de 6 articulações não é quadrada, tornando impossível determinar as combinações singulares através do cálculo do seu determinante.

$$\begin{bmatrix} J_v \\ J_w \end{bmatrix} = \begin{bmatrix} J_{v1} & J_{v2} & J_{v3} & J_{v4} \\ J_{w1} & J_{w2} & J_{w3} & J_{w4} \end{bmatrix} \tag{4.22}$$

Genericamente, uma singularidade pode ser verificada através do Jacobiano, quando uma dada combinação de valores implica a existência colunas dependentes, tornando o sistema indeterminado. Para qualquer manipulador, independentemente do número e tipo de articulações que o constituem, é possível encontrar tais combinações recorrendo à matriz pseudo-inversa do Jacobiano ($J^+ = J^T(JJ^T)^{-1}$), e verificando para que condições o determinante de JJ^T se anula (39).

O cálculo do Jacobiano foi assim feito recorrendo à ferramenta Matlab. Dado o tamanho das matrizes calculadas ser demasiado grande, assim como as equações para o determinante de JJ^T, resultado da presença de quatro articulações rotativas, estes resultados não foram incluídos nesta secção, sendo no entanto possível consultar o código desenvolvido no anexo B.1. Todavia, substituindo na equação $det(JJ^T)$ o valor $\theta_4 = 0º$, este determinante anula-se, confirmando a existência de uma singularidade neste ponto. Da mesma forma, substituir $\theta_2 = \pm 90º$ anula igualmente este determinante e corresponde, portanto, à segunda singularidade deste manipulador. Efectivamente, colocar $\theta_2 = \pm 90º$ corresponde à abertura dos braços para os lados, paralelamente ao chão, alinhando as articulações associadas a θ_1 e θ_2, e originando a perda de independência de colunas do Jacobiano deste manipulador e, portanto, a diminuição da sua característica.

Assim, foram adicionadas duas condições (equações 4.23 e 4.24) com o objectivo de eliminar a instabilidade dos ângulos θ_1 e θ_3. Note-se que, embora $\theta_2 = -90^\circ$ origine igualmente uma singularidade, esta não foi considerada, uma vez que esta articulação do robô está fisicamente limitada, sendo impossível atingir este valor.

$$\theta_1 = \begin{cases} \theta_1 & se|\theta_2 - 90^\circ| > 5^\circ \\ 0^\circ & se|\theta_2 - 90^\circ| \leq 5^\circ \end{cases} \tag{4.23}$$

$$\theta_3 = \begin{cases} \theta_3 & se|\theta_4| > 5^\circ \\ 0^\circ & se|\theta_4| \leq 5^\circ \end{cases} \tag{4.24}$$

4.4 Geração de Trajectórias das Pernas

A matriz H_{foot}^{waist} resultante da equação 4.21 permite relacionar as posições e orientações do tronco e do pé (direito) em função dos valores dos seis ângulos aplicados a cada uma das articulações que os unem. Desta matriz, as três primeiras colunas reflectem a orientação relativa do pé, enquanto que a quarta consiste na sua posição, expresso nas coordenadas do referencial associado

ao tronco[4]. O processo inverso – o da determinação dos ângulos que validam esta relação – é a etapa que sucede ao conjunto de passos discriminados na secção 2.1.1, e que descreve o algoritmo de locomoção implementado na versão mais recente do robô NAO. Neste processo, e uma vez que as origem das três articulações da bacia coincidem, $c = d = 0$ (figura 4.6). Também as origens das duas articulações do tornozelo são coincidentes, pelo que foi considerado $g = 0$. As dimensões das restantes variáveis poderão ser consultadas no anexo C.

4.4.1 Método Baseado em Análise Numérica

As coordenadas do tronco e do pé, extremidades de cada perna, estão separadas por uma diferença cuja representação segundo os três eixos cartesianos pode ser expressa segundo qualquer um dos dois referenciais, dependendo se se recorre à matriz H_{foot}^{waist} ou H_{waist}^{foot}. Assim, é possível conhecer uma equação para cada eixo, expressa em função dos seis ângulos $\theta_1, ..., \theta_6$. Contudo, a existência de três equações linearmente independentes, aliada à existência de múltiplas singularidades, aumenta grandemente o número de soluções que este sistema admitiria. Por outro lado, o carácter altamente não linear de cada uma destas equações elimina qualquer hipótese de recorrer aos métodos tradicionais de análise inversa da cinemática, pelo elevado esforço computacional que tal exigiria do processador. Relembre-se assim que o objectivo é proporcionar um algoritmo que permita gerar um movimento de locomoção rápido e estável.

A literatura contempla vários algoritmos utilizados para contornar as dificuldades acima mencionadas (como por exemplo o método do ponto fixo, ou o método de Newton) (42). Estes consistem geralmente em processos iterativos que visam provocar alterações nas várias entradas e avaliar os respectivos efeitos nas saídas. Numa dada iteração, uma alteração é considerada válida quando o efeito que produz na saída é favorável à satisfação de um ou vários critérios (como por exemplo a diminuição da diferença entre o resultado desejado e o resultado da alteração). O restante desta secção sugere a implementação de um algoritmo que procura determinar um ou vários conjuntos de seis ângulos que satisfaçam a relação entre o pé e o tronco.

4.4.1.1 Parâmetros

Procurou-se implementar um sistema versátil e genérico, cuja qualidade do desempenho estivesse confinada apenas à escolha de parâmetros mais ou menos adequados. Assim, o algoritmo é executado a partir dos parâmetros abaixo indicados.

- **N Ângulos antes da actuação** *(vector Nx1)*, que constituem o ponto de partida do algoritmo
- **Passo mínimo** e **passo máximo**, que quantificam as perturbações mínima e máxima a aplicar a cada ângulo. O valor realmente aplicado é determinado através de uma função quadrática, limitada por estes dois coeficientes

[4]Dado o tamanho da matriz que resulta desta multiplicação matricial, o seu resultado simbólico não se encontra directamente explícito neste documento, sendo possível determiná-lo pela execução em Matlab do algoritmo apresentado no anexo B.2

- **Agressividade do passo** (explicado na secção 4.4.1.3)
- **Limites das articulações** *(vector Nx2)*, para que a solução determinada pelo algoritmo seja válido não só em ambiente de simulação como também quando aplicada no robô
- **Posição relativa desejada do tronco** ***(target)*** *(vector 3x1)*, expresso no referencial associado ao pé de suporte, e sobre o qual se pretende determinar os ângulos
- **Número máximo de iterações** do algoritmo
- **Erro máximo** entre a posição desejada do tronco e a posição obtida numa dada iteração, e que termina o algoritmo

4.4.1.2 Algoritmo

O algoritmo proposto para a determinação dos ângulos foi desenvolvido em Matlab, tendo sido procurado estruturar o código de forma versátil e genérica. Desta forma é possível e fácil a sua reutilização em outros manipuladores, independentemente do número e tipo de articulações que o constituem, uma vez que recorre exclusivamente às equações que relacionam as coordenadas cartesianas de dois referenciais, em função dos valores actuados nas articulações que os unem. As figuras 4.7 e 4.8 reflectem a implementação desse mesmo algoritmo.

Uma função de alto nível, *main()*, carrega os parâmetros a partir dos quais todo o processamento matemático é executado, invoca funções de mais baixo nível e, quando finalizado o processamento, devolve os valores determinados para os ângulos, bem como um histórico da evolução da posição ao longo das várias iterações.

A função *convLoop()*, de médio nível, executa uma rotina cíclica que, a cada iteração, determina uma nova combinação de seis ângulos, tal que cada um, quando o seu efeito é avaliado de forma independente, constitui um mínimo local do erro. O erro é determinado como sendo a soma das diferenças absolutas entre a posição actual e a desejada do tronco, expresso em coordenadas do referencial do pé. No fim de cada iteração, esta função avalia quão longe está a solução obtida da solução procurada e, mediante a satisfação ou não de um determinado critério de paragem (secção 4.4.1.4), termina o algoritmo, ou inicia uma nova iteração.

Por fim, *convIT()* constitui, paralelamente a *fwdKinematics()*, a função de mais baixo nível, descrevendo o processamento matemático que caracteriza uma iteração. Este é constituído por uma etapa inicial, em que algumas variáveis são guardadas, seguida de uma rotina cíclica, que é executada tantas vezes quanto o número de articulações do manipulador. Este ciclo consiste, inicialmente, pela determinação da tendência da função. Este valor determina qual deverá ser a influência do passo (positivo ou negativo) que contribuirá para a aproximação da posição do manipulador ao ponto desejado e, consequentemente, da descoberta do mínimo local. A tendência (t na figura) será 1 ou -1, significando que a distância ao ponto desejado é menor do que a actual, quando se incrementa ou decrementa, respectivamente, a quantidade definida pelo passo. Descoberto este valor, a função progride (dentro dos limites das articulações) até que o mínimo local seja encontrado.

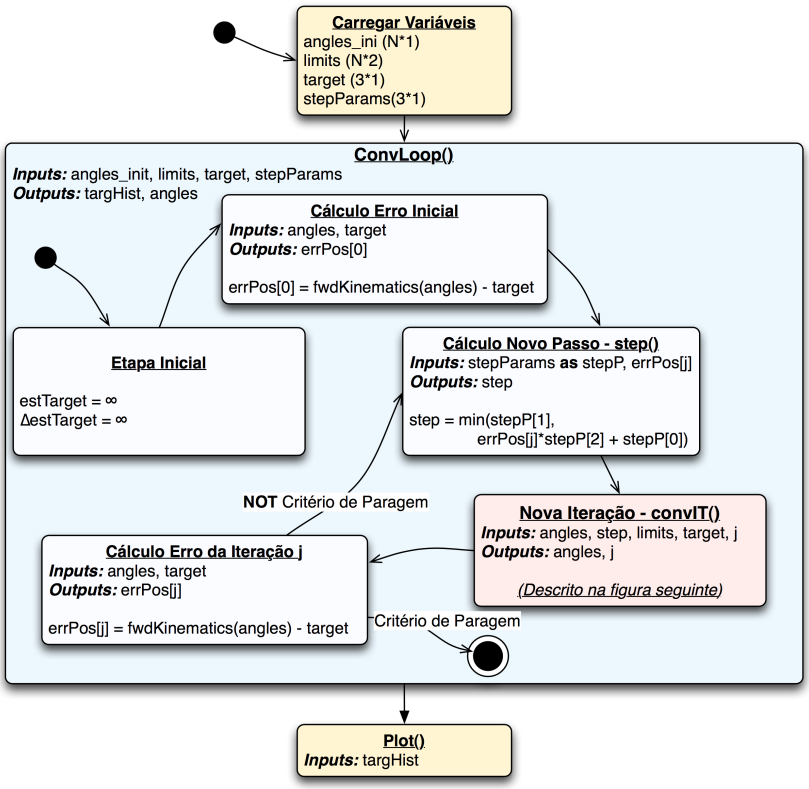

Figura 4.7: Algoritmo de determinação de ângulos de um manipulador, dadas as coordenadas da extremidade

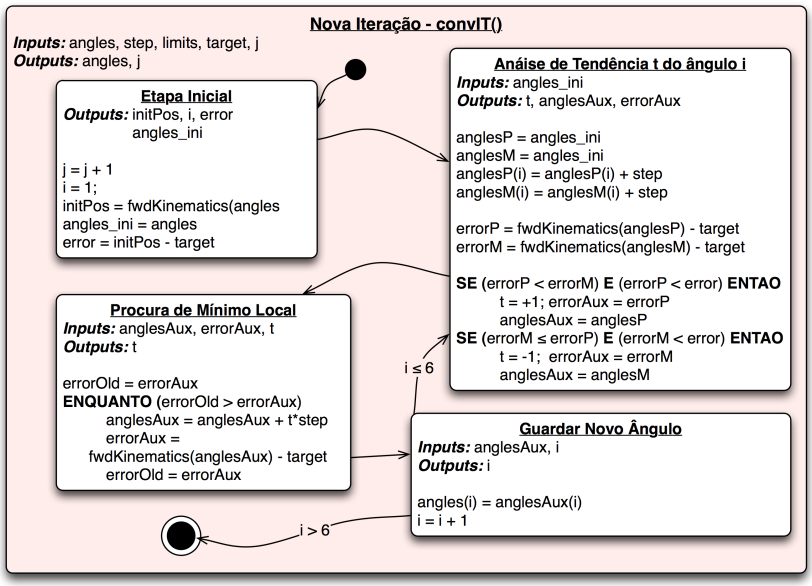

Figura 4.8: Algoritmo de determinação de ângulos de um manipulador, dadas as coordenadas da extremidade (continuação)

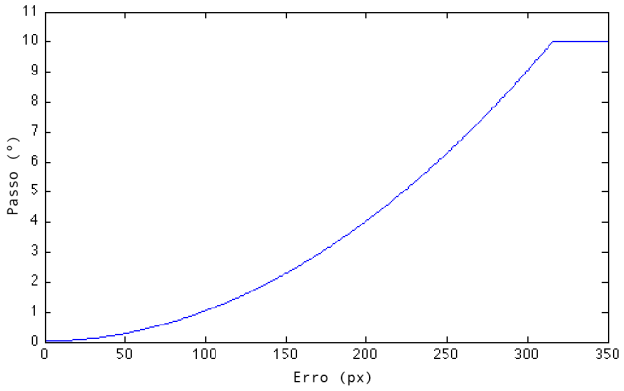

Figura 4.9: Relação entre o erro de posição do tronco e o passo do algoritmo

4.4.1.3 Passo Variável

O passo consiste na perturbação introduzida em cada um dos ângulos ao longo do algoritmo, cujo efeito é gerar deslocações do tronco. No entanto, enquanto que um passo pequeno pode traduzir-se num aumento substancial do número de cálculos efectuados em cada iteração, um passo demasiado elevado pode implicar a assunção de pontos perto de mínimos locais como se deles se tratasse, resultando numa solução final mais afastada da posição desejada (caso o algoritmo pare pelo alcance do número limite de iterações), ou num aumento do número de iterações, dado que cada uma contribui de forma mais lenta na obtenção da solução. Em suma, um passo grande é favorável quando, tipicamente no início do algoritmo, os ângulos iniciais se encontram afastados da solução desejada, enquanto que um passo pequeno é adequado quando o erro é menor, a fim de garantir uma solução de forma mais resoluta.

Assim, optou-se por utilizar um passo variável em função do erro, de forma a utilizar o valor mais adequado em cada circunstância. Este consiste numa equação de segunda ordem, limitada superior e inferiormente, através de parâmetros especificados antes da execução do programa. A abertura da parábola, que é típica das equações de segundo grau, é também arbitrada no parâmetro *agressividade do passo* (a). A equação 4.25 e a figura 4.9[5] reflectem assim esta relação entre o passo ($step$) e o erro calculado (e).

$$step(e) = \begin{cases} a.e^2 + step_{min} & se\ step(e) \leq step_{max} \\ step_{max} & se\ step(e) > step_{max} \end{cases} \tag{4.25}$$

4.4.1.4 Critério de Paragem

O critério de paragem varia de acordo com os parâmetros que são indicados na execução do programa tendo sido, para o efeito, consideradas duas situações: o número de iterações já executadas, ou o erro máximo admissível entre as posições desejada e a obtida numa iteração.

[5] $a = 0.0001$; $step_{min} = 0.05$; $step_{max} = 10$

Assim, a condição de paragem, que na execução do programa é representada por uma variável boleana, é considerada verdadeira quando pelo menos uma das duas condições se verifica.

Capítulo 5

HumaNAO

5.1 Introdução

HumaNAO consiste numa aplicação desenvolvida no âmbito desta Dissertação e que visa, de acordo com os objectivos propostos na secção 1.3, propor um mecanismo inovador de interacção entre o ser humano e o robô NAO. O principal objectivo desta aplicação é a capacidade de enviar ordens de actuação para o NAO, à distância, sendo estas executadas no mesmo instante.

Teleoperação – a arte de controlar robôs à distância – não é uma novidade. Efectivamente, já vários estudos foram feitos em robôs, com o objectivo de encontrar novas maneiras de os controlar remotamente, ou providenciar mecanismos para os ensinar por demonstração. Em tais investigações constam sempre os objectivos de aumentar o realismo da interacção homem-robô, ou o desenvolvimento de sistemas de captação de movimentos menos invasivos e com maior liberdade de movimentos (43; 44; 45).

Apesar deste conceito ser aplicável a qualquer robô em geral, a teleoperação em robôs humanóides está ainda subdesenvolvida. Contudo, e visto um humanóide possuir uma estrutura física semelhante à dos humanos, poder controlar remotamente um robô com sucesso constitui uma vantagem no domínio do ensino por demonstração. Em 1998, S. Lee *et al.* (43) desenvolveu um método de teleoperação baseado em movimentos aplicados sobre um braço mecânico acoplado ao braço humano. Os ângulos aplicados sobre as articulações podem ser facilmente determinados através da análise da cinemática inversa. Por outro lado, a presença de uma malha de realimentação permite ao operador experienciar as mesmas forças (ou equivalentes proporcionais) aplicadas ao robô, através de actuadores pneumáticos. Um mecanismo idêntico também foi aplicado para as articulações da mão.

Entretanto, desde o uso de periféricos de entrada de um computador (tais como um rato ou um teclado) (46), *joysticks* (44; 47) até dispositivos com ainda mais graus de liberdade e menos restrições de movimento, muitas outras soluções surgiram. Em particular, foi sugerido um controlador baseado numa miniatura de robô (45), cujos movimentos são intuitivamente transferidos para o real. O processo inverso foi também considerado: uma vez o robô em operação, a miniatura poderia ser usada para reflectir os movimentos do robô real. Já em 2010, Y. Lu *et al.* (48) sugeriu

o uso da voz para controlar o robô humanóide BHR-2, cujos comandos, uma vez compreendidos por um algoritmo próprio, invocariam rotinas previamente memorizadas.

O ensino por demonstração baseado em teleoperação é uma área conhecida, mas em evidente crescimento. Assim, este capítulo apresenta uma nova proposta de ensino por demonstração usando teleoperação, recorrendo à câmara Kinect.

5.1.1 Requisitos

A primeira etapa do desenvolvimento da aplicação, fulcral para o sucesso de todas as etapas subsequentes, consistiu numa análise criteriosa de todos os requisitos e restrições que este sistema deveria possuir.

- O sistema deverá ser capaz de reconhecer e seguir os movimentos de um ser humano;
- O sistema de reconhecimento deverá ser não-invasivo;
- O robô NAO deverá executar as ordens recebidas em menos de 0.2 segundos;
- O sistema deverá permitir o seguimento de várias pessoas (uma de cada vez), sem implicar a reinicialização das aplicações;
- O robô NAO deverá receber ordens através de uma interface sem fios;
- O sistema deverá tornar possível controlar um robô à distância;
- Deverá ser possível executar a aplicação local a partir de qualquer computador que corra uma distribuição do Linux;
- Deverá ser possível executar a aplicação em qualquer robô NAO;
- Deverá ser minimizado o esforço computacional do robô;

5.1.2 Aplicações

A robótica é uma ciência em constante desenvolvimento. Contudo, embora o controlo de robôs industriais à distância seja um assunto já abordado e constante da literatura (49), o controlo de robôs humanóides por mimetização e sem recurso a tecnologias invasivas é ainda um tema pouco explorado, e cujo potencial contributo científico é reconhecido. Efectivamente, uma vez criada tecnologia que permita a aquisição de tais sistemas a preços competitivos, facilmente se reconhece um número elevado de aplicações para ela. Embora seja mais facilmente perceptível a utilidade de tais robôs na preservação da segurança das pessoas, existem outras áreas onde estes podem actuar. Seguem-se abaixo alguns exemplos:

- **Altas temperaturas:** salvamento de pessoas dentro de edifícios em chamas;
- **Áreas submersas:** investigação de áreas de profundidade elevada, sem recurso a oxigénio, e sem limitações de pressão da água;

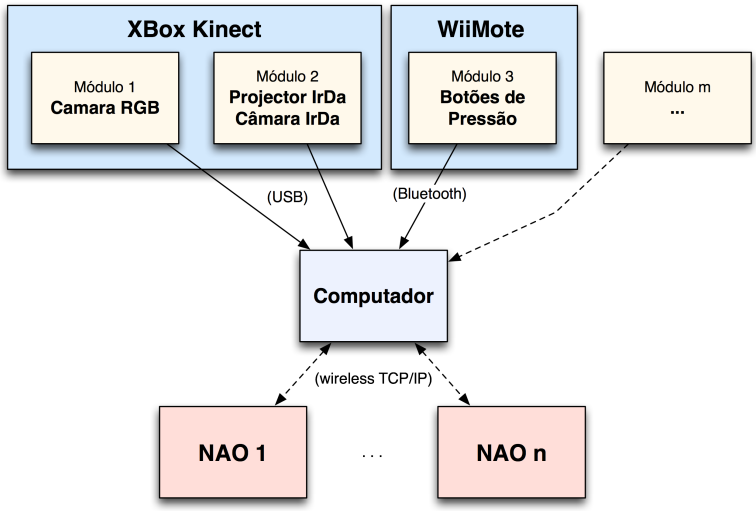

Figura 5.1: Arquitectura de *hardware* de alto nível

- **Interacção segura:** aproximação segura a grupos maliciosos (ladrões, terroristas);
- **Vigilância à distância:** supervisão e interacção à distância com menores, em casa;
- **Conforto:** execução de tarefas domésticas.

5.2 Arquitectura da Aplicação

Aquando da especificação da solução, foi tomada em consideração a possibilidade de expansão futura do sistema, quer pela adição de novas funcionalidades, quer pela inclusão de outras tecnologias que pudessem interagir com o robô. Deste modo, as figuras 5.1 e 5.2 reflectem respectivamente as arquitecturas de *hardware* e *software* do sistema, e que visam satisfazer todos os requisitos impostos na secção 5.1.1.

Assim, o sistema recorre ao seguinte *hardware*:

- **XBox Kinect:** Câmara da Microsoft XBox 360, com capacidade para capturar a cor e profundidade de píxeis numa janela 640x480;
- **Wiimote:** Comando da Nintendo Wii
- **Computador:** Processador com grande capacidade de processamento, interface de comunição sem-fios e interface USB (utilizada pela Kinect);
- **Router:** Dispositivo gerenciador de comunicações dentro de uma rede;
- **Robô NAO**

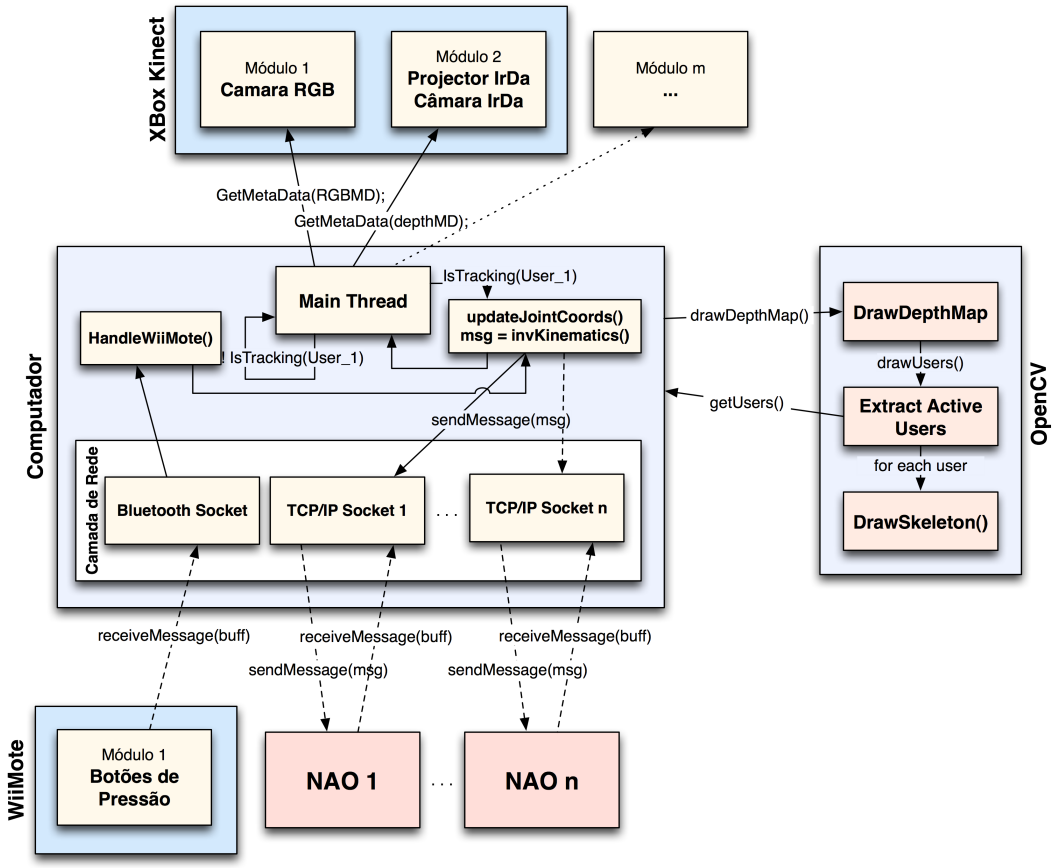

Figura 5.2: Arquitectura de *software* de alto nível

O restante deste capítulo procura explicar em detalhe todo o processo de aquisição e processamento das informações provenientes do Kinect, bem como o mecanismo de actuação implementado no robô NAO. A implementação da solução proposta foi feita por etapas, criando e testando pequenas aplicações antes do seu agrupamento numa única aplicação.

5.3 Aplicação Local

A aplicação local consiste num programa desenvolvido em C, cujo principal objectivo é servir de interface entre o Kinect (via USB), o comando Wiimote (via *Bluetooth*) e o robô NAO (via rede *ethernet* sem-fio), permitindo também a este último libertar grande parte do seu esforço computacional para o computador onde esta aplicação corre. A aplicação local, no contexto deste projecto, é responsável pela execução de várias tarefas, abaixo discriminadas:

- Aquisição da imagem
- Detecção de novas pessoas
- Seguimento de pessoas detectadas
- Determinação de ângulos das articulações dos braços baseados na análise da cinemática inversa da primeira pessoa detectada;
- Identificação e actuação de acções esporádicas sobre o controlo do robô, enviadas pelo comando Wiimote
- Manutenção de uma (ou mais) ligação sem fios
- Manutenção de um registo histórico em formato legível no Matlab de todas as informações relativas às posições das articulações do corpo humano, captadas pelo Kinect, para posterior análise e eventual depuração de erros

Com esta aplicação pretende-se demonstrar habilidades do NAO através do controlo dos braços à distância, utilizando o Kinect para captar os movimentos de uma pessoa, enquanto que o comando Wiimote será utilizado como mecanismo de segurança através da actuação dos vários botões, ou para a realização de outras funções, convenientemente referidas ao longo deste documento. Procurou-se criar um algoritmo simultaneamente robusto, não invasivo, e de resposta rápida.

5.3.1 Execução Sequencial *versus* Execução Paralela

Quando aplicações, pelas suas características ou finalidades, implicam a introdução de conceitos de tempo-real, torna-se necessário considerar uma arquitecta de processamento adequada. As seguintes estruturas são utilizadas (50):

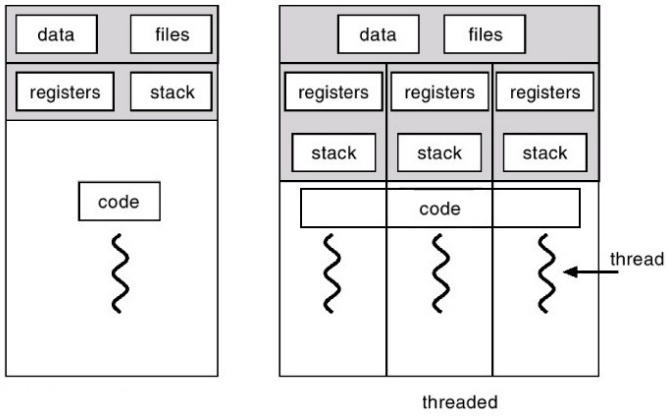

Figura 5.3: Execução de várias tarefas em simultâneo numa mesma aplicação

- **Execução cíclica:** as tarefas são realizadas pela mesma ordem, sendo que uma tarefa só é executada depois que todas as tarefas antes dessa tenham acabado a sua execução. Esta é a estrutura de mais fácil implementação e entendimento. No entanto, não garante o cumprimento de restrições temporais.

- **Execução baseada em interrupções:** as tarefas são executadas aquando da geração dos respectivos eventos, permitindo libertar o processador para outros processos. No entanto, a ocorrência de eventos enquanto o processador está em utilização por esse processo pode levar à sua perda.

- **Execução em paralelo:** várias tarefas são executadas virtualmente *em simultâneo* pelas unidades de processamento. Tal mecanismo implica a divisão de uma linha de execução – *thread* – em duas ou mais linhas de execução distintas. *Threads* consistem nas unidades de processamento de menor dimensão que podem ser agendadas por um sistema operativo, que processa partes de cada uma alternadamente.

Dado se ter procurado especificar um sistema expansível, garantindo o cumprimento de restrições temporais de tarefas de maior importância, foi adoptado um modelo de execução em paralelo no desenvolvimento da aplicação, de acordo com a figura 5.4. Este modelo permite recorrer à biblioteca *pthread.h*, constante das bibliotecas padrão dos sistemas operativos Unix, que contém já um conjunto de funções e classes necessárias à criação de novos processos e tarefas (*threads*), e a mecanismos que permitem à aplicação gerir de que forma o acesso ao processador é feito. Assim, o processo compreende diferentes funções, que são executadas como *threads* independentes:

- **main:** Inicialização de variáveis e tarefas utilizadas ao longo do programa, bem como da câmara Kinect;

- **glutDisplay:** Actualização das matrizes de cor e profundidade provenientes da Kinect, e da consola OpenCV;

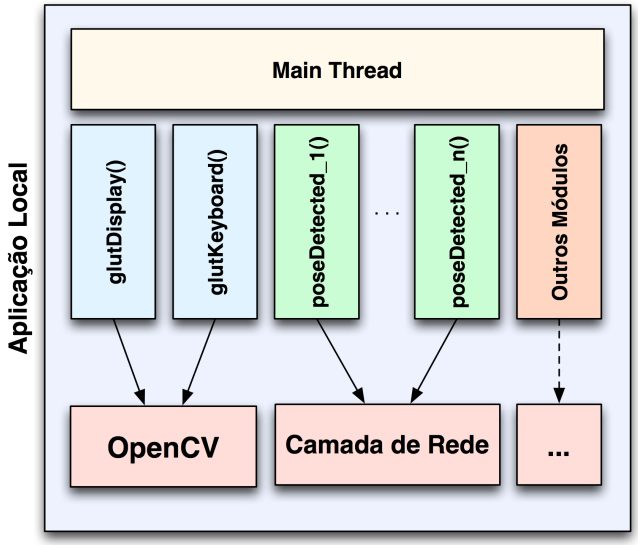

Figura 5.4: Arquitectura de *software* da aplicação local

- **glutKeyboard:** Monitorização de eventos recebidos através do teclado;
- **poseDetected_x:** Monitorização do esqueleto da pessoa *x*. Actualização das coordenadas das articulações. Cálculo da cinemática inversa para detecção dos ângulos das articulações. Concatenação das informações numa trama. Envio da trama por TCP/IP.
- **wiiComm_tf:** Gere os eventos gerados pelo comando Wiimote associado à aplicação.

5.3.2 OpenNI

O processo de aquisição de imagens de cor e profundidade a partir da Kinect é feito com recurso à framework *OpenNI*[1], criada em 2010 por uma empresa com o mesmo nome e com o objectivo de oferecer um conjunto de APIs e bibliotecas *open-source* que permitissem estandardizar e simplificar o acesso a dispositivos de interacção natural, como o Kinect.

Uma vez que o software de instalação é acompanhado de alguns exemplos, a utilização desta plataforma tornou possível a simplificação de algumas etapas cujo desenvolvimento não é relevante do contexto deste projecto, nomeadamente as que respeitam ao processamento de imagem e detecção de esqueletos provenientes da análise da informação proveniente da câmara. Em particular, a aplicação *Sample-NiUserTracker* (37) contém um conjunto de funções e instruções que permitem não só essa detecção, como também a sua monitorização, até que as pessoas detectadas desapareçam do plano da câmara.

[1] http://www.openni.org

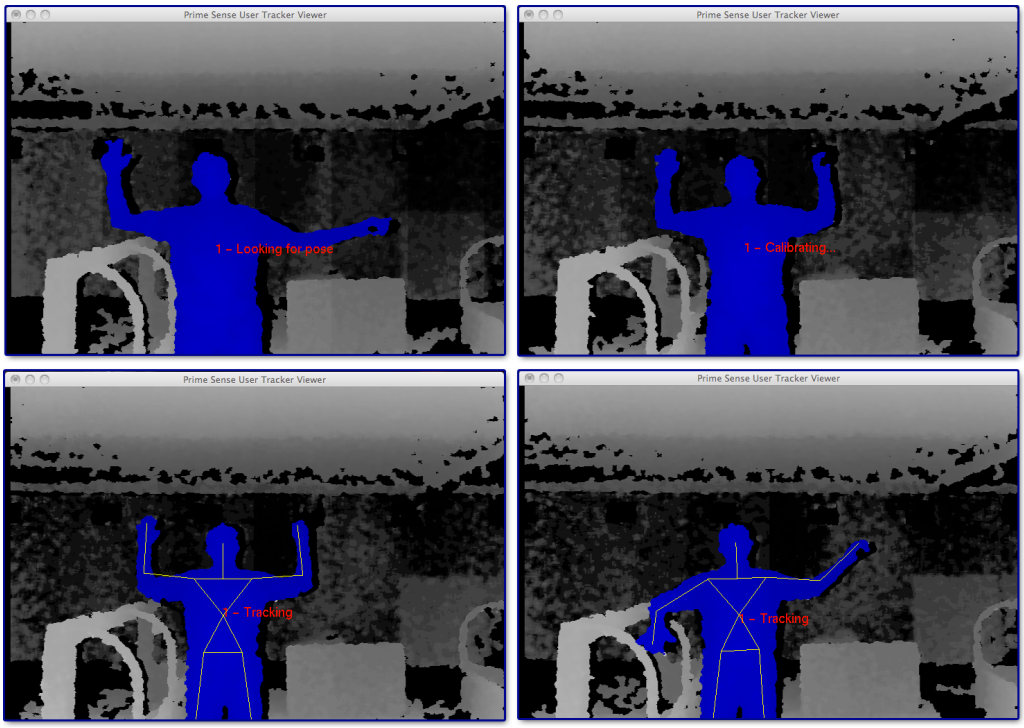

Figura 5.5: Etapas do algoritmo de detecção e seguimento de um esqueleto da *OpenNI*

5.3.3 Algoritmo de Calibração

Para que seja possível o correcto seguimento de um esqueleto de um ser humano, é necessário um algoritmo de calibração robusto que permita a identificação inicial da pessoa. As bibliotecas disponibilizadas pela *OpenNI* contêm já um conjunto de funções que permitem este reconhecimento. Este mecanismo recorre à matriz de profundidades enviada pela Kinect, e a cada nova imagem identifica quais os pontos cujo valor se alterou. Desta forma é possível segmentar todo o conjunto de corpos em movimento, e que representam potenciais pessoas. A figura 5.6 representa o processo de detecção e seguimento de corpos implementado e que compreende, para cada corpo detectado, quatro etapas:

- **Procura de pose:** O programa segue o corpo até este executar uma pose com a forma do caracter grego psi (ψ). Uma vez detectado, o corpo transita para o estado de calibração.

- **Calibração:** O programa aguarda que o corpo pare o seu movimento, para registar as coordenadas iniciais das articulações. Finda esta etapa, o programa inicia a monitorização do corpo.

- **Monitorização:** O programa executa uma função que, de forma periódica, actualiza as coordenadas de todas as articulações do corpo.

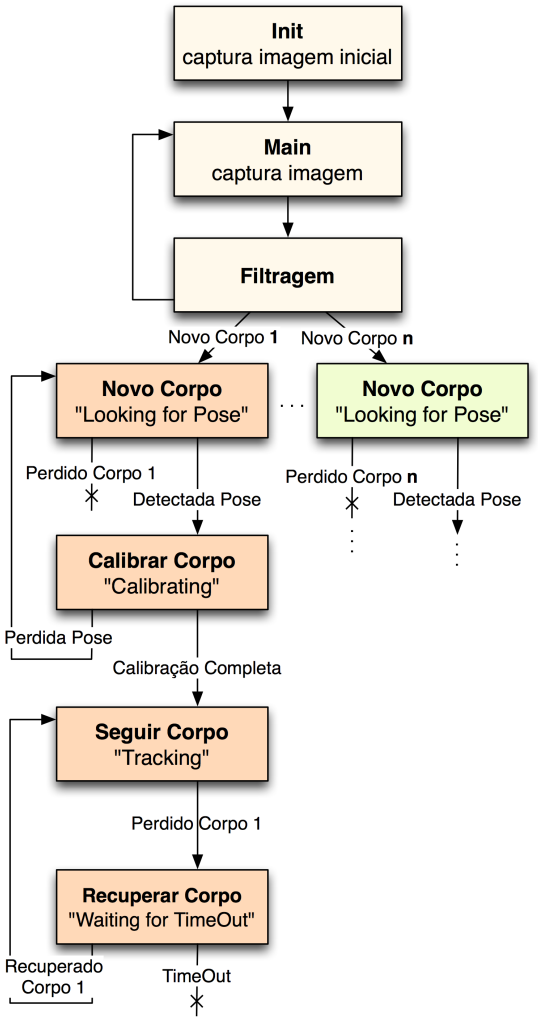

Figura 5.6: Algoritmo de detecção e seguimento de corpos da *OpenNI*

- **Recuperação:** Uma vez perdido um corpo em monitorização, o programa aguarda que este volte a aparecer dentro de um limite temporal pre-estabelecido, a partir do qual o registo do corpo é apagado, e o processo de reconhecimento reiniciado.

5.3.4 Filtro Passa-Baixo

O Kinect foi utilizado ao longo do projecto para identificar e devolver as coordenadas das várias articulações do corpo humano, sendo cada um destes valores armazenado numa estrutura a que foi dado o nome *NaoPoint*. Um sistema que associa directamente uma dada referência (ou um valor proporcional a ela) a uma saída é designado como um sistema em malha-aberta (51). Por oposição, um sistema em malha fechada tira partido do estado actual em busca de uma resposta estável para sistemas que por defeito não são estabilizáveis quando controlados por uma malha aberta de controlo. De facto, uma vez que em malha aberta a saída depende exclusivamente da

entrada que lhe é aplicada, esta carece de mecanismos que permitam avaliar se tal informação corresponde à realidade, ou se foi involuntariamente alterada pela existência de perturbações. Tais interferências são geralmente modelizadas como um ruído branco[2], impedindo que seja possível determinar uma frequência ou amplitude para estes valores (52). Assim, o controlo em malha-fechada possibilita a utilização de mecanismos que visam corrigir ou atenuar ruídos aleatórios que afectam a qualidade da referência.

Com efeito, a informação devolvida pelo Kinect – uma matriz de profundidades e uma matriz de cores – apresentam algum ruído que, na utilização em video-jogos da XBox 360 podem ser desprezáveis, mas que no âmbito deste projecto, em que é pretendido controlar os movimentos de um robô, são significativos. Facilmente se entende que a origem de tais erros pode estar associada às condições de iluminação de uma sala, à textura e material dos corpos captados (afectando o sinal infra-vermelhos reflectido em cada ponto) ou mesmo à qualidade e resolução da electrónica utilizada na construção da câmara. Como tal, foi implementado um filtro do tipo passa-baixo discreto, que visa atenuar as componentes de maior frequência. A fim de poupar esforço computacional, o filtro foi aplicado não à totalidade das matrizes, mas apenas nas coordenadas identificadas como correspondendo às articulações do esqueleto (num total de 14).

A literatura refere uma grande variedade de filtros, que variam na sua complexidade, esforço computacional e por fim no efeito atenuante na referência. Nesta aplicação foi usado um filtro de primeira ordem, discretizado através do método de diferenciação regressiva de Euler (53). A transformada de Laplace da função de transferência deste filtro é dada pela equação 5.1.

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{Ts+1} \tag{5.1}$$

em que T, $y(s)$ e $u(s)$ são, respectivamente, a constante de tempo, a saída e entrada do filtro. Uma vez a multiplicação por s no domínio das frequências representa uma derivação no domínio dos tempos, a equação pode ser rescrita de acordo com a equação 5.2, em que t_k representa um instante de tempo discretizado.

$$Tsy(s) + y(s) = u(s) \quad \overset{\mathscr{L}^{-1}}{\longleftrightarrow} \quad T\dot{y}(t_k) + y(t_k) = u(t_k) \tag{5.2}$$

Na equação acima, $\dot{y}(t_k)$ é determinado pelas equações 5.3 e 5.4.

$$\dot{y}(t_k) \approx \frac{y(t_k) - y(t_{k-1})}{h} \tag{5.3}$$

$$h = t_k - t_{k-1} \tag{5.4}$$

Assim, o filtro passa-baixo discreto pode ser finalmente determinado como dependente da taxa de amostragem (h) e da constante de tempo (T) através das equações 5.5 e 5.6. α é chamado de *parâmetro do filtro*.

[2] $v(s)$ e $v(t)$ independentes para qualquer $t \neq s$

$$y(t_k) = \alpha y(t_{k-1}) + (1-\alpha)u(t_k) \tag{5.5}$$

$$\alpha = \frac{T}{T+h} \tag{5.6}$$

O valor de h é conhecido, e consiste no intervalo de tempo que separa cada nova aquisição de imagem por parte da Kinect. Este valor foi determinado empiricamente, através da comparação de valores devolvidos em cada iteração pela função *gettimeofday()*, da biblioteca *sys/time.h*, tendo-se obtido um valor médio aproximado de $32ms$.

Por outro lado, para evitar que o efeito do filtro passa-baixo discreto se afaste do resultado que adviria da aplicação do seu equivalente contínuo (equação 5.1), o valor da taxa de amostragem h deverá ser significativamente menor do que a constante de tempo T, numa razão de, pelo menos, cinco vezes (53) ($h \leq T/5$). O valor de T, maior do que $160ms$, pode ser determinado com base na frequência de corte desejada para o filtro, igualando o denominador da equação 5.1 a zero (equação 5.7).

$$Ts + 1 = 0 \leftrightarrow 2\pi f = \frac{1}{T} \leftrightarrow f = \frac{1}{2\pi T} \tag{5.7}$$

O gráfico da figura 5.7 representa a relação entre a frequência de corte f e os valores que o parâmetro do filtro passa-baixo α toma (equação 5.6), bem como a constante de tempo T, resultantes da variação deste valor.

5.3.5 Camada de Rede

A camada de rede consiste num conjunto de funções que visam assegurar a comunicação entre duas ou mais aplicações, e a sua configuração e funcionamento são assegurados através de bibliotecas específicas de cada sistema operativo, onde se encontram definidas as *sockets* das várias interfaces de comunicação disponíveis para cada máquina. *Sockets* são terminais de comunicações bi-direccionais (ponto-a-ponto) cujo objectivo é gerir o fluxo de informação que chega à maquina, redirecionando-a para os processos aos quais tal informação se destina.

Afim de criar uma ligação com sucesso, foi importante considerar quais os requisitos que a camada de rede deveria obedecer, a saber:

- A qualidade e fiabilidade da informação transmitida deverão ser assegurados;
- Cada transmissão de informação deverá demorar menos de 50 ms;
- O protocolo deverá permitir estabelecer-se uma comunicação num ambiente sem-fios, suportado tanto pelo NAO como pela máquina onde a aplicação executa.

O robô NAO, conforme constituído actualmente, suporta apenas comunicações via *Ethernet*, *Bluetooth* e *Wi-Fi*. Por uma questão de versatilidade, e uma vez que a maioria dos computadores

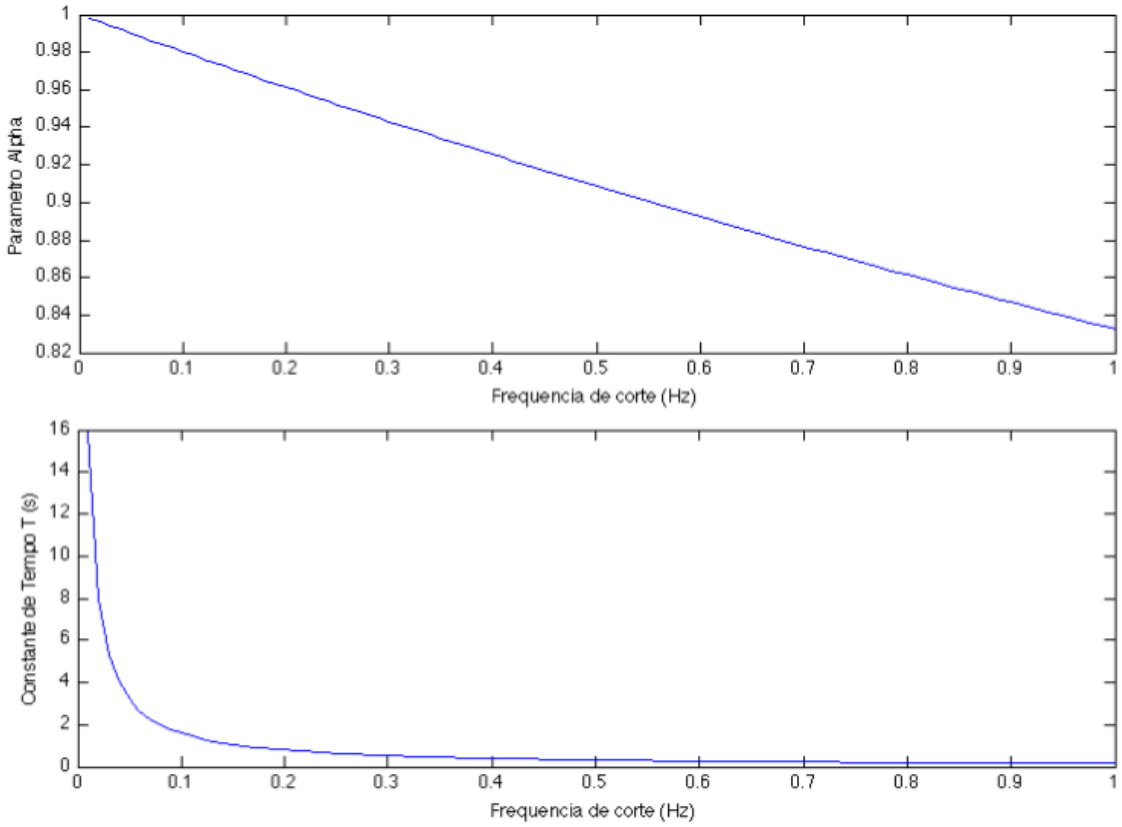

Figura 5.7: Influência da frequência f na sintonização do filtro passa-baixo discreto – parâmetro α *(acima)* e constante de tempo T *(abaixo)*

suporta comunicações através de uma rede local, optou-se por recorrer a um protocolo de comunicação IP instalado numa rede local, comum aos dois dispositivos. Por outro lado, este permite também que com pequenas modificações seja possível controlar o robô via *Internet*.

O protocolo de comunicação IP, ao nível da camada de rede, define de que modo as mensagens devem ser divididas, e define mecanismos para que estas possam ser reorganizadas no destino. Entre outros dados, esta camada contem informações sobre qual o remetente e destinatário de uma mensagem, bem como a que protocolo da camada seguinte (camada de transporte) esta deve ser entregue. Actualmente a tecnologia oferece dois protocolos para a camada de transporte:

- **TCP[3]/IP:** Mecanismo de transmissão complexo, que realiza uma transmissão por segmentos enumerados. O fluxo de dados é controlado, garantindo que não são perdidos pacotes da mensagem.
- **UDP[4]/IP:** Mecanismo simples e de implementação pouco complexa. É rápido, mas não possui controlo de fluxo, podendo levar à perda de pacotes. É adequado a tráfego cuja qualidade de serviço não é uma exigência.

Uma vez que é crucial que o envio correcto das mensagens seja assegurado, neste projecto foi utilizado o protocolo TCP/IP. Também, e uma vez o fluxo de ordens é unilateral, da aplicação local para o NAO, a ligação segue um padrão de comunicação do tipo "Mestre/Escravo". O mestre é a aplicação local – de onde as ordens são enviadas – e o escravo é o NAO que, recebendo novas ordens, as extrai da mensagem e as executa (54).

A utilização de um protocolo de comunicação estandardizado permite que dois dispositivos possam comunicar entre si com recurso a bibliotecas que a maior parte dos sistemas operativos já inclui por defeito. Desta forma, a utilização do protocolo TCP/IP permite que o envio de mensagens seja bem sucedido pela inclusão de vários parâmetros de controlo que transparecem aquando desenvolvimento de aplicações (figura 5.8).

A camada de rede, responsável pela troca de informações com outros dispositivos (neste caso o robô NAO) é composta por um número variável de *sockets*, cujo objectivo é estabelecer linhas de comunicação ponto-a-ponto, sendo a configuração de cada uma das *sockets* feita imediatamente após o arranque da aplicação. O processo de execução de uma comunicação, tal como descrito na figura 5.9, é iniciado pela instanciação de uma nova *socket*, seguida da sua associação ao segundo dispositivo, através do seu endereço IP, e da sua porta. No final da execução do programa, a ligação é terminada, e a *socket* destruída.

5.3.5.1 Formação da Mensagem

A aplicação e o robô comunicam entre si através do canal TCP/IP explicado na secção 5.3.5. Através dele são enviadas mensagens que, consoante o primeiro caractér, podem assumir diferentes funções:

[3]Transmission Control Protocol
[4]User Datagram Protocol

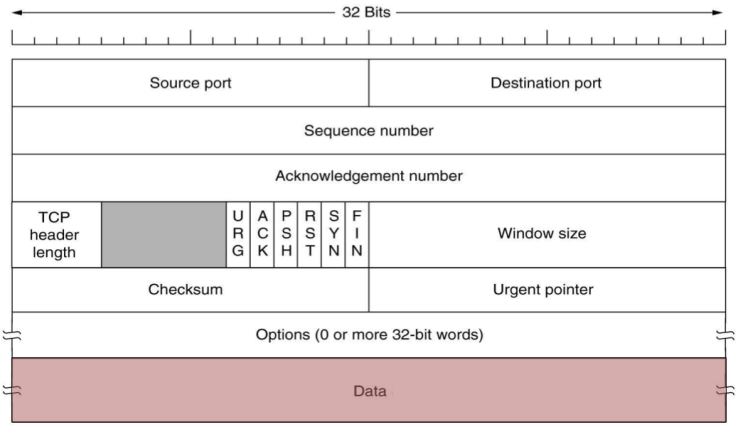

Figura 5.8: Trama TCP/IP (54)

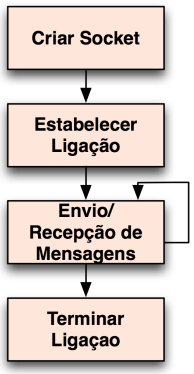

Figura 5.9: Diagrama de execução de uma *socket* de um cliente

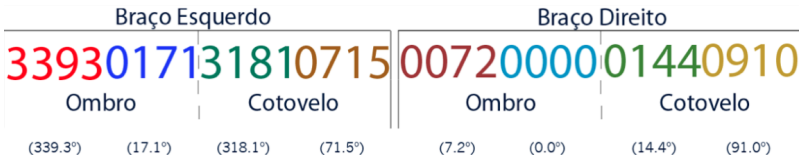

Figura 5.10: Mensagem enviada para o NAO

- **'$':** Os três caracteres seguintes definem um novo valor para o coeficiente do filtro passa-baixo discreto aplicado ao nível dos actuadores do NAO. Aquando do arranque do programa, a mensagem *'$000'* é automaticamente enviada.

- **'#':** Activação ou desactivação do modo de registo dos valores das articulações num ficheiro Matlab. Esta funcionalidade é utilizada para limitar o informação útil ao longo da execução do processo, diminuindo o tamanho dos respectivos ficheiros gerados. Ao primeiro caractér – '#' – sucede apenas um segundo, podendo tomar dois valores possíveis – '0' indica um pedido de criação de um novo ficheiro (ou sobreposição se tal já existir), e '2' indica um pedido de encerramento do mesmo ficheiro, adicionando os caracteres necessários para que este possa ser correctamente executado no Matlab.

- **Outros caracteres:** A mensagem consiste no conjunto dos ângulos provenientes da análise da cinemática, com a resolução de 0.1º, sendo cada ângulo sempre representado com 4 dígitos e numa gama de 0000 a 3600, e cujo valor é determinado segundo o algoritmo descrito na secção 4.3. A figura 5.10 representa um exemplo de uma mensagem completa, em que cada caracter corresponde a um dígito dos vários ângulos.

5.4 Aplicação Remota – NAO

A execução da aplicação foi concebida para ser dividida entre duas máquinas. Enquanto que o restante deste capítulo descreveu o processamento que cabia à primeira máquina – um computador com elevada capacidade de processamento – esta secção descreve a aplicação desenvolvida para o NAO. Dada a velocidade do processador deste ser substancialmente menor (500MHz (55)), procurou-se tirar o maior partido possível do computador, tentando que a informação que é transmitida entre as duas máquinas careça do menor volume de tratamento possível no robô.

5.4.1 Arquitectura de *Hardware* e *Software*

O maior volume de processamento no robô NAO é realizado por um processador AMD Geode 500MHz com 256Mb de memória SDRAM[5], situado na cabeça. Ainda, é neste processador que o sistema operativo (Linux) é executado, bem como o controlador do NAO *NaoQi* (explicado

[5] *Synchronous Dynamic Random Access Memory*

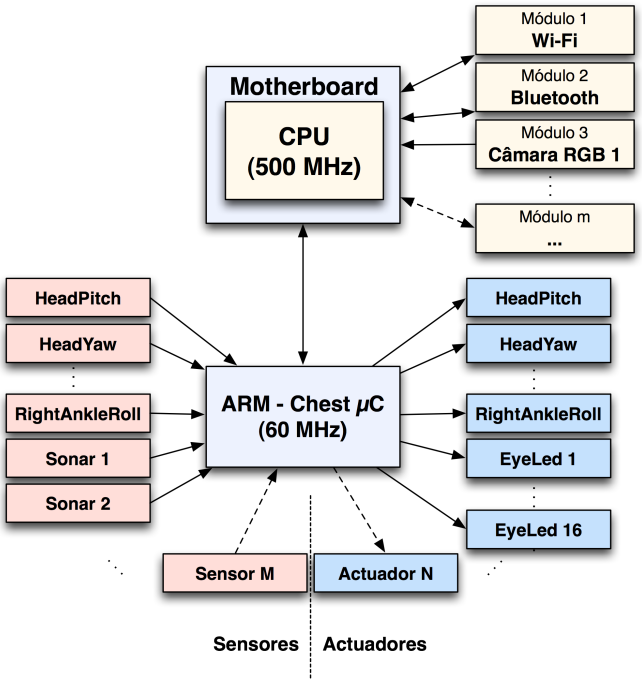

Figura 5.11: Arquitectura de *hardware* de alto nível do robô NAO

adiante) e ainda outras tarefas determinadas pelo utilizador. À *motherboard* onde este processador pertence está ligado um microprocessador ARM7 de 60MHz, situado no tronco e que, numa camada de mais baixo nível, faz a interface entre a *motherboard* e cada um dos microcontroladores associados a cada um dos actuadores do robô. Além disso, é o ARM que recolhe toda a informação sensorial ao longo de todo o corpo, não só a que respeita aos valores das várias articulações como também aos sensores de ultra-som, acelerómetros, botões de pressão, entre outros. Apenas os sensores e actuadores instalados na cabeça são uma excepção, estando directamente ligados à *motherboard*. Na figura 5.11 é possível perceber a arquitectura de *hardware* de alto nível implementada no robô NAO.

NaoQi consiste no controlador de mais alto nível que descreve a arquitectura de *software* do NAO. Este controlador pode ser executado directamente a partir do robô, remotamente a partir de um computador, ou de forma distribuída entre estes. Este controlador está dividido em três partes distintas, conforme descritas abaixo.

- **NaoQi Daemon:** corre os processos principais do sistema operativo. No caso de se pretender aceder aos sensores e actuadores do NAO, o *NaoQi Daemon* deve ser executado no robô.

- **NaoQi Library:** oferece um conjunto de bibliotecas e funcionalidades que permitem a chamada de algoritmos e funções já implementadas no NAO, permitindo que o controlo do robô seja feito sem aceder directamente ao *hardware*.

- **Device Control Manager (DCM):** permite o acesso directo ao *hardware*, por comunicação directa com o microcontrolador situado no tronco. O controlo do robô por intermédio do DCM permite explorar o *hardware* de forma mais eficiente, e aumentar o desempenho do robô. Este método, no entanto, não tira partido do controlo implementado pela Aldebaran Robotics nos restantes módulos do *NaoQi Library*, pelo que os mecanismos existentes por defeito para o controlo de estabilidade encontram-se desactivados, cabendo ao programador providenciar os seus próprios mecanismos.

A programação em ambiente *NaoQi* é orientada ao objecto, e o desenvolvimento de aplicações para o NAO passa pela criação de módulos (os objectos de mais pequena dimensão). A troca de informações entre os vários módulos é feita por intermédio de um outro módulo, de nome *ALMemory*. Este é responsável pela gestão de uma memória partilhada, por a disponibilizar e actualizar, a pedido de outros módulos. Entre várias outras informações, a memória partilhada contém um registo actualizado das informações provenientes dos sensores e as ordens de actuação enviadas para as várias articulações, pertinentes neste projecto. O acesso a este módulo está no entanto confinado a módulos constantes da *NaoQi Library* e do DCM.

A aplicação desenvolvida no âmbito deste projecto tira partido do DCM. Em causa está o controlo dos braços do robô em tempo-real e em simultâneo com o ser humano que se coloca por trás do Kinect, fazendo com que o domínio da estabilidade não seja um desafio. Relembre-se pelo exposto na secção 2.1.1 que o robot manterá a sua estabilidade sempre que projecção do seu centro de massa no plano do chão estiver dentro do polígono convexo definido pelos apoios (pés) do robô.

Desde Setembro do ano transacto que a FEUP e o IEETA[6] têm cooperado mutuamente no estudo do robô NAO, com vista ao desenvolvimento da equipa de futebol robótico. Um dos resultados do trabalho destas duas instituições consiste num módulo de particular interesse – *dcmcontroller*. Em conjunto com todos os dependentes deste, este módulo permite simplificar o acesso ao *hardware* do NAO, por interacção directa com o DCM, e com a memória partilhada.

Assim como a aplicação local utiliza uma *socket* que permite estabelecer uma ligação pontual com outro dispositivo, também o NAO possui uma *socket*, que constitui a outra extremidade desta via de comunicação. Esta no entanto opera como um servidor, na medida em que o robot, aquando da sua inicialização, entra automaticamente num modo de escuta, permanecendo neste modo até que algum outro dispositivo entre em contacto com ele (o que será feito no início da execução da aplicação no computador).

A figura 5.12 descreve a arquitectura de *software* do NAO, e de que maneira o módulo desenvolvido nesta aplicação interage com os restantes. A execução deste módulo inicia-se com a bifurcação do processo em duas *threads*, uma das quais se destina a gerir o fluxo de informação através da *socket* TCP/IP, enquanto que a outra executa uma rotina cíclica que visa ordenar a actuação dos ângulos recebidos da aplicação local nas articulações. Conforme a figura demonstra, a segunda *thread* consiste numa execução cíclica das seguintes quatro funções:

[6] Instituto de Engenharia Electrónica e Telemática de Aveiro

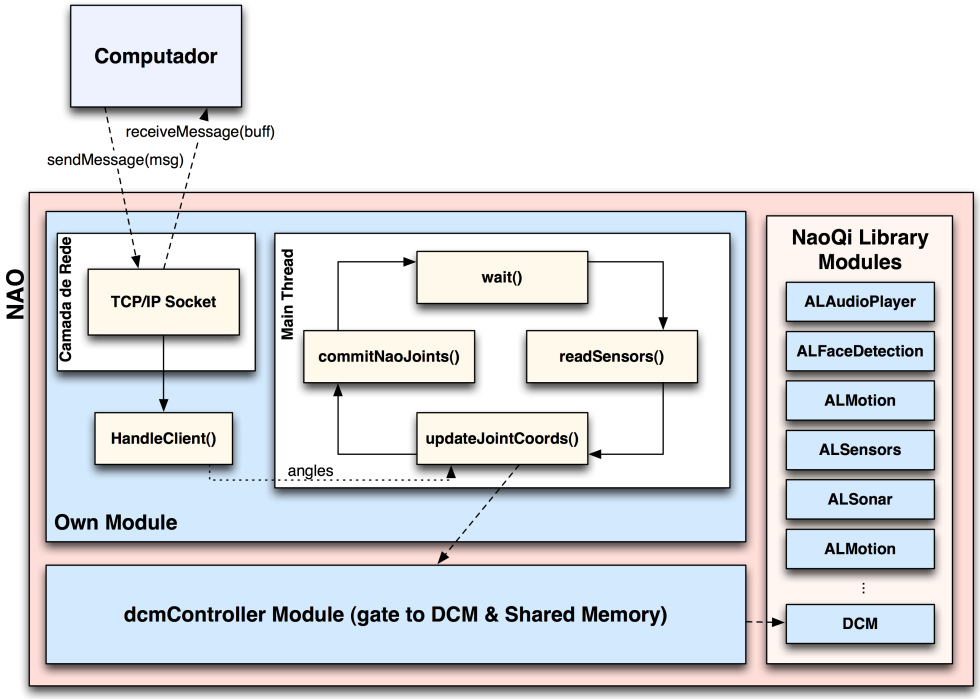

Figura 5.12: Arquitectura de *software* do robô NAO

- **wait():** como o próprio nome indica, a memória partilhada é um recurso utilizado por vários módulos ao longo da execução de um processo do NAO. A gestão de acessos à memória é feita com recurso a um semáforo, sendo o objectivo da função *wait()* impedir a evolução do processo enquanto o acesso não tiver sido alocado a este módulo.

- **readSensors():** actualiza toda a informação sensorial relevante à aplicação.

- **updateJointCoords():** converte a informação mais actual dos ângulos recebidos pela rede de graus para radianos, convertendo para o seu simétrico quando necessário. O resultado desta função consiste num vector que contém os ângulos a aplicar nas articulações, depois filtrados por um filtro passa-baixo (caso o seu coeficiente seja diferente de zero). É também nesta função que um ficheiro é preenchido (legível em Matlab), contendo um registo completo dos ângulos recebidos pela rede – referências – e os ângulos reais das articulações.

- **commitNaoJoints():** escreve os valores dos ângulos determinados na função anterior na memória partilhada. É nesta função que o robô assume uma posição definida pelo conjunto de ângulos que recebeu.

O anexo A.2 contém uma descrição sumária das principais funções implementadas neste módulo.

5.5 Ensino por Demonstração

A possibilidade de controlar um robô ao mesmo tempo que as respectivas ordens são executadas permite facilmente perceber um vasto domínio de aplicações onde este conceito pode ser aplicado. O mesmo acontece quando se procura munir um robô de técnicas e procedimentos que possam ser reproduzidos na íntegra, em qualquer instante, um qualquer número de vezes. Esta possibilidade é uma mais valia actualmente em desenvolvimento, com especial incidência na automatização de processos de produção, com vista à substituição de mão de obra humana por tecnologia que reproduza movimentos repetitivos. Do processo resultará uma produção de forma continuada, mais rápida e mais precisa.

Neste projecto procurou-se transportar este conhecimento para o robô humanóide NAO e, tirando partido das potencialidades da câmara Kinect e do NAO, criar uma estrutura que permitisse armazenar rotinas, identificadas por um nome. Assim, execução do programa segue por um de três algoritmos possíveis. Um primeiro – *Live* – permite o controlo livre do robô, em tempo-real, sem registo de movimentos. O algoritmo *NewRecord* inicia uma rotina cuja execução é acompanhada em tempo-real, pelo NAO, mas cujos movimentos são enviados paralelamente para o NAO e para uma base de dados. Por fim, a rotina *UseRecord* é utilizada para aceder à base de dados e, chamando o ficheiro respectivo, envia sequencialmente os movimentos nele contidos para o NAO, através da rede. Este envio é feito a uma cadência semelhante à das duas outras rotinas, pelo que foram utilizadas funções constantes das bibliotecas do sistema que permitem pausar o desenvolvimento de determinadas funções durante um período de tempo estipulado.

5.5.1 Argumentos de Invocação da Aplicação

O acesso às rotinas é feito aquando da invocação do programa, através dos argumentos definidos na linha de comandos. Desta forma, esta invocação é feita através do comando *./SampleNiEE06161 <endereço> <porta> <param1> [nome_do_programa W/R]*.

- **endereço:** endereço IP do robô NAO *(obrigatório)*.
- **porta:** porta por onde esta aplicação troca informações *(obrigatório)*.
- **param1:** coeficiente inicial do filtro passa-baixo, descrito na secção 5.3.4 *(obrigatório)*.
- **nome_do_programa:** nome do ficheiro contendo sequências de movimentos que se pretende executar ou criar *(facultativo)*.
- **W/R:** especificação do modo de operação – *W* ou *w* cria ou sobrepõe um novo algoritmo (modo *NewRecord*), *R* ou *r* executa no robô determinado pelo *endereço* o algoritmo com o nome *nome_do_programa* (modo *UseRecord*) *(facultativo, excepto quando nome_do_programa é definido. A não definição destes parâmetros implica a execução do algoritmo Live)*.

Figura 5.13: Comando Wiimote

5.5.2 Comando Wiimote

O Wiimote foi desenvolvimento pela Nintendo, para a consola Nintendo Wii, e consiste num comando sem-fios através do qual é possível controlar um video-jogo. Ao contrário de marcas concorrente, o comando desenvolvido por esta empresa possui uma ergonomia que permite manuseá-lo com apenas uma mão. Por outro lado, e uma vez que o Wiimote comunica com a consola Wii por intermédio de uma rede *Bluetooth*, um protocolo de comunicação estandardizado, facilmente se encontra na *Internet* uma vasta oferta de bibliotecas que permitem ligar o comando a um computador.

Na sua constituição, o comando integra doze botões de pressão (figura 5.13), e ainda quatro LEDs, uma câmara infra-vermelhos, 3 acelerómetros, um altifalante, um motor de vibração e uma interface para conectar outros controladores (56).

No contexto desta dissertação, procurou-se utilizar o comando com vista a proporcionar métodos de interacção de forma mais expedita com a aplicação. Por outro lado, o Wiimote foi utilizado também como dispositivo de segurança. Assim, foram implementadas funções em resposta às seguintes combinações de botões:

- **Setas (↑↓):** inicia e termina, respectivamente, o registo de um ficheiro Matlab, contendo informações sobre os ângulos extraídos da mensagem recebida pela rede, em comparação com os valores actuais das articulações.
- **A:** congela a renovação dos ângulos, mantendo o robô estático numa dada posição;
- **B:** sobrescreve a mensagem enviada para o robô NAO, qualquer que ela seja, para uma sequência de 32 zeros, obrigando o robô a esticar ambos os braços para a frente;
- **A + B:** termina a execução do programa;
- **-/+:** decrementa/incrementa o coeficiente do filtro passa-baixo em 0.01 .

5.5.3 Base de Dados

A manutenção de uma base de dados permite armazenar e gerir as várias sequências de movimentos criadas. De forma a manter a aplicação versátil, e a fim de poupar esforço computacional, procurou-se recorrer a um mecanismo leve em tamanho, de rápido acesso, e que não carecesse de *software* adicional para o seu acesso. Assim, a cada rotina é associado um ficheiro de texto,

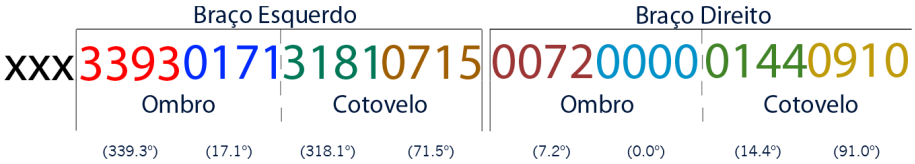

Figura 5.14: Mensagem escrita no ficheiro de texto do algoritmo de ensino por demonstração

cujas linhas reflectem novas combinações dos 8 ângulos a aplicar, dispostos de forma semelhante à exemplificada na figura 5.10, mas cujas mensagens são precedidas de três dígitos adicionais, cuja composição indica o número de repetições dessa linha (figura 5.14). A linha de execução de alto nível segue, portanto, um processamento representado pela figura 5.16.

As etapas *Processar Mensagem* e *Escrever Nova Linha* recebem como parâmetros a mensagem a enviar, e um vector de boleanos que caracteriza o estado dos botões do comando Wiimote, permanentemente actualizados por uma função executada em paralelo com o resto do programa. Assim, uma combinação particular dos botões do comando gera, no instante de enviar a mensagem para o robô NAO, um dos comportamentos descritos pelo algoritmo descrito na figura 5.15.

Note-se ainda na figura 5.16 que os modos *NewRecord* e *Live* enviam, sempre que o esqueleto é perdido (pela saída da pessoa do campo de visão da câmara) uma última mensagem, que consiste numa sequência de 32 zeros. Esta mensagem é enviada por uma questão de segurança, uma vez que, desconhecida a última posição captada da pessoa, esta poderia eventualmente levar o robô a executar movimentos que pudessem danificar a sua estrutura. A mensagem *"00000000000000000000000000000000"* permite ao robô, em caso de controlo inactivo, colocar-se numa posição segura até nova ordem de controlo.

```
i = 0;
Mensagem_Anterior = '00000000000000000000000000000000';

while NOT(Botao_A AND Botao_B)
    i = i + 1;

    if NOT Botao_A
        Mensagem_Nova = Calcular_Cinematica_Inversa();
    else if Botao_B
        Mensagem_Nova= '00000000000000000000000000000000';
    end if

    if NOT(Mensagem_Anterior = Mensagem_Nova)
        Ficheiro = i + Mensagem_Anterior;
        NAO = Mensagem_Nova;
        Mensagem_Anterior = Mensagem_Nova;
        i = 0;
    end if
end while
```

Figura 5.15: Algoritmo de geração de mensagens

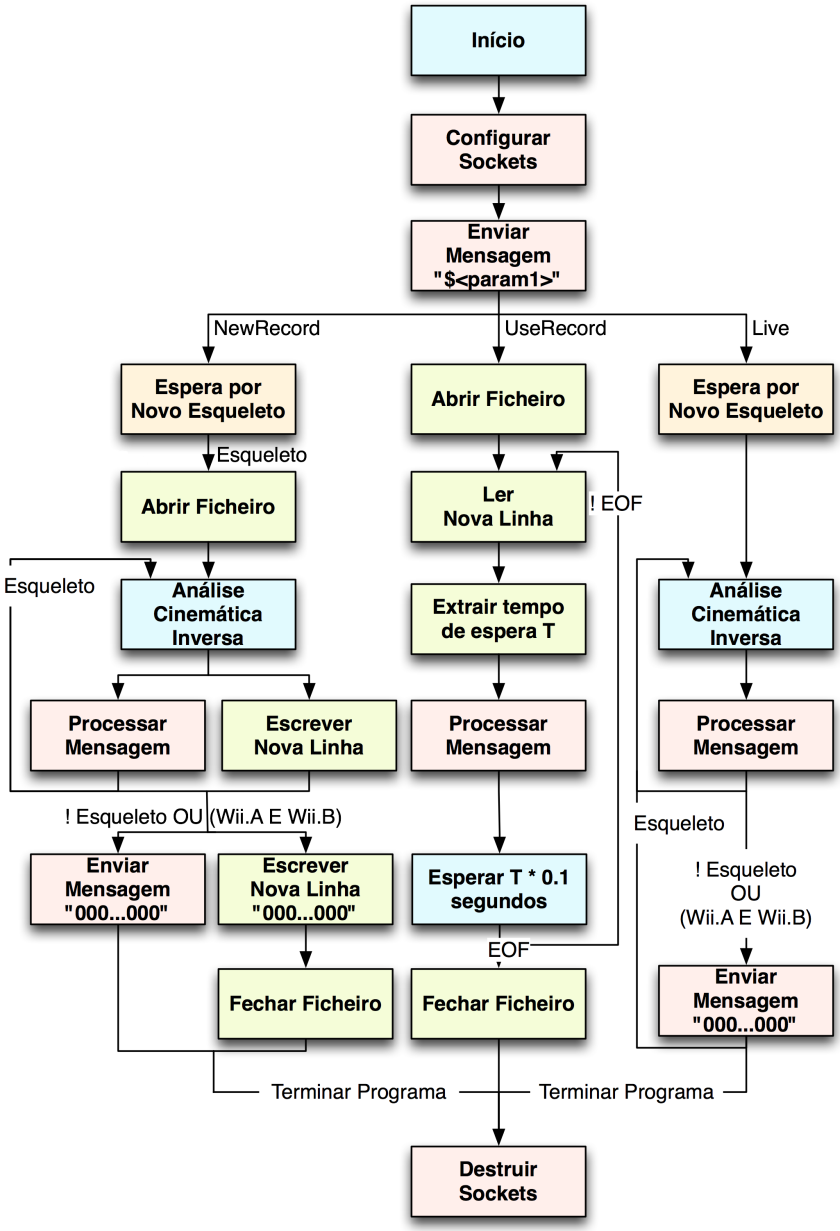

Figura 5.16: Modelo de execução de alto nível da aplicação

Capítulo 6

Resultados e Validação da Solução

Este capítulo visa descrever os principais testes realizados sobre o sistema, a fim de averiguar o seu comportamento, e verificar se a solução satisfaz ou não os requisitos identificados no início do projecto. Os vários testes incidem sobre a configuração inicial da aplicação, o tempo de resposta do NAO face aos movimentos realizados à frente da câmara Kinect, o efeito dos botões do comando Wiimote no funcionamento da aplicação, a geração de ângulos das pernas na geração de trajectórias e ainda a execução de algoritmos previamente gravados, no âmbito do ensino por demonstração.

Os testes foram feitos recorrendo à consola do sistema operativo, ao *tcpServSim* e *tcpCliSim*[1] e ao Matlab. A validade de cada uma das etapas da solução é feita com base em várias *checklists*, que definem os objectivos que se pretendem ver atingidos.

6.1 Arranque da aplicação

Este primeiro teste pretende avaliar se as configurações iniciais dos vários dispositivos (NAO, Wiimote e aplicação local) são efectuadas com sucesso, e se as várias ligações ponto-a-ponto são criadas e operam sem falhas. Além da configuração do Kinect, o início da aplicação consiste na criação das ligações, no envio de uma primeira mensagem do computador para o NAO ('$000'), cujo objectivo é desligar o filtro passa-baixo dos actuadores, e na activação das articulações do NAO, de forma a este produzir força suficiente nos motores para estar de pé, com os braços na horizontal.

As tabelas 6.1 e 6.2 definem as *checklists* de verificação das funcionalidades da aplicação.

6.2 Kinect e extracção de coordenadas

Os testes realizados no âmbito do Kinect relacionam-se com a frequência com que todo o processo de captura da imagem, extracção das coordenadas das articulações, e a determinação dos

[1] Programas criados em C++ para emular um servidor TCP/IP (semelhantes ao que é executados no NAO e na aplicação local)

Configuração das Ligações		
#	**Descrição**	**Resultado**
1.1.1	Imediatamente após o arranque, a aplicação local envia um pedido de acesso ao NAO	✓
1.1.2	O NAO envia uma resposta à aplicação local	✓
1.1.3	Imediatamente após o arranque, a aplicação do Wiimote envia um pedido de acesso à aplicação local	✓
1.1.4	A aplicação local envia uma resposta à aplicação do Wiimote.	✓

Tabela 6.1: *Checklist* de configuração das ligações

Arranque da Aplicação		
#	**Descrição**	**Resultado**
1.2.1	O NAO recebe a mensagem inicial '$000'	✓
1.2.2	O NAO coloca-se numa posição de equilíbrio, com os braços esticados	✓
1.2.3	Perante os argumentos invocados, a aplicação evolui para o rotina *Live*	✓
1.2.4	Perante os argumentos invocados, a aplicação evolui para o rotina *NewRecord*	✓
1.2.5	Perante os argumentos invocados, a aplicação evolui para o rotina *UseRecord*	✓
1.2.6	A invocação da rotina *UseRecord* inibe a iniciação da câmara Kinect	✓
1.2.7	O NAO não aceita mais do que uma ligação em simultâneo.	✓

Tabela 6.2: *Checklist* de arranque da aplicação

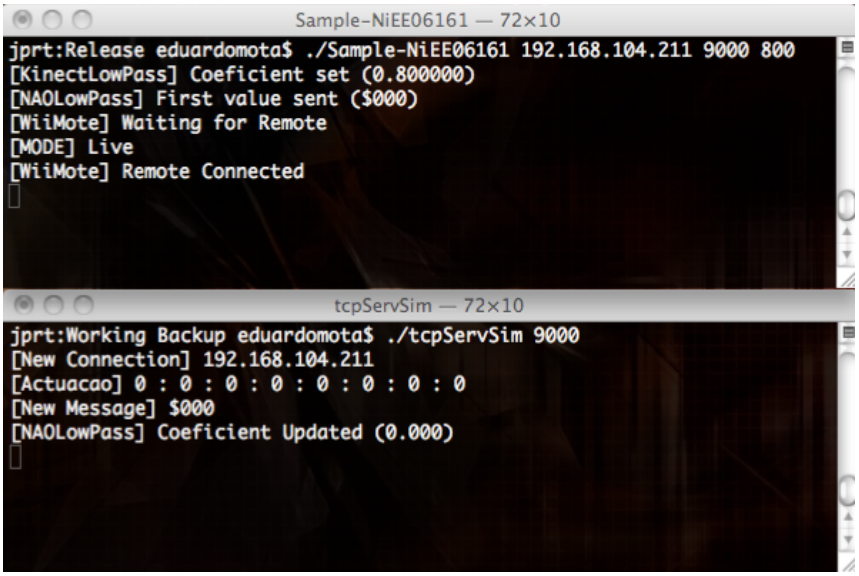

Figura 6.1: Arranque da aplicação e configuração das ligações

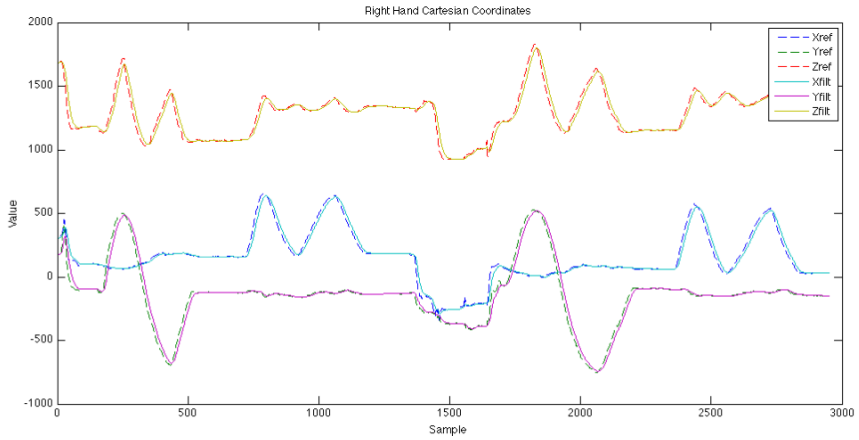

Figura 6.2: Efeito do filtro passa-baixo sobre os valores recebidos do Kinect

ângulos através da cinemática inversa é feito. Este valor representa a maior parte do atraso entre a realização de um movimento, e a sua repetição pelo NAO. Por outro lado, a introdução do filtro passa-baixo visa suprimir o ruído introduzido pela câmara, com o custo de implicar uma resposta tanto mais lenta quanto maior for o valor do coeficiente deste (até o máximo de 1.000, exclusive). A figura 6.2 ilustra o efeito do filtro passa-baixo implementado sobre os valores recebidos do Kinect, para um coeficiente de 0.909 – o que corresponde a uma frequência de corte de 0.5 Hz – dentro da gama admissível que este parâmetro pode tomar (secção 5.3.4). O resultado da aplicação do filtro é visível na suavidade com que as trajectórias são descritas pelo NAO, e na redução substancial do ruído, responsável pela instabilidade de controlo antes evidente. A figura 6.3 evidencia o sucesso do filtro na eliminação desse mesmo ruído.

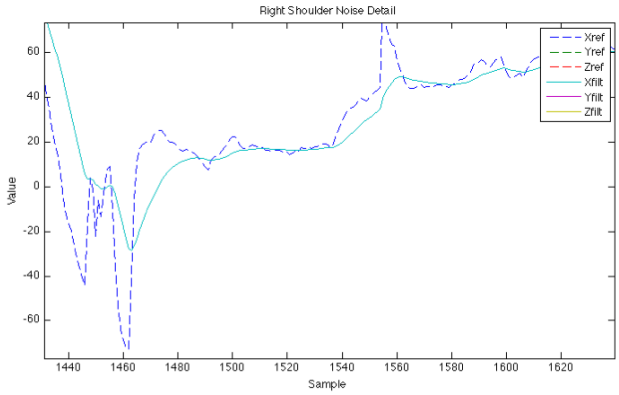

Figura 6.3: Pormenor do efeito do filtro-baixo na eliminação de ruído de alta frequência

KINECT E EXTRACÇÃO DE COORDENADAS		
#	**Descrição**	**Resultado**
2.1.1	Os ângulos das articulações são actualizados a uma frequência maior do que 25 Hz (superior à frequência de envio de mensagens na rede)	✔
2.1.2	O filtro passa-baixo encontra-se correctamente configurado, respeitando os limites temporais impostos	✔
2.1.3	A aplicação mantém um histórico completo das coordenadas extraídas do Kinect, antes e após a sua passagem pelo FPB, legível em Matlab	✔
2.1.4	Os ângulos calculados descrevem com rigor os movimentos realizados à frente da câmara	✔

Tabela 6.3: *Checklist* do Kinect e extracção de coordenadas

6.3 Geração de Trajectórias das Pernas

A descoberta de uma combinação de seis ângulos que coloquem a projecção do tronco dentro da área abrangida por um pé é apenas uma das etapas do processo de geração de trajectórias das pernas. Este teste visa averiguar se o algoritmo sugerido consegue determinar tal combinação, e se esta permite à posição do tronco convergir para a posição desejada. O algoritmo, tal como descrito na secção 4.4 foi testado recorrendo a uma simulação em Matlab. Afim de poder testar a eficiência do algoritmo face a condições iniciais variadas, os ângulos iniciais consistem em seis valores aleatórios, numa gama de -180º a +180º (figura 6.4).

A tabela 6.4 reflecte os resultados de uma simulação que foi executada cinco mil vezes, permitindo testar o algoritmo com um número suficiente de amostras, e cuja simulação pudesse ser processada num período de tempo razoável. Foram considerados diferentes passos, partindo da mesma combinação de ângulos iniciais (gerados novamente em cada iteração). Sabe-se que existe pelo menos uma combinação de ângulos que satisfaz a posição desejada do tronco (neste caso foi utilizada a posição alcançada quando todos os ângulos têm o valor 0º). Considera-se ainda

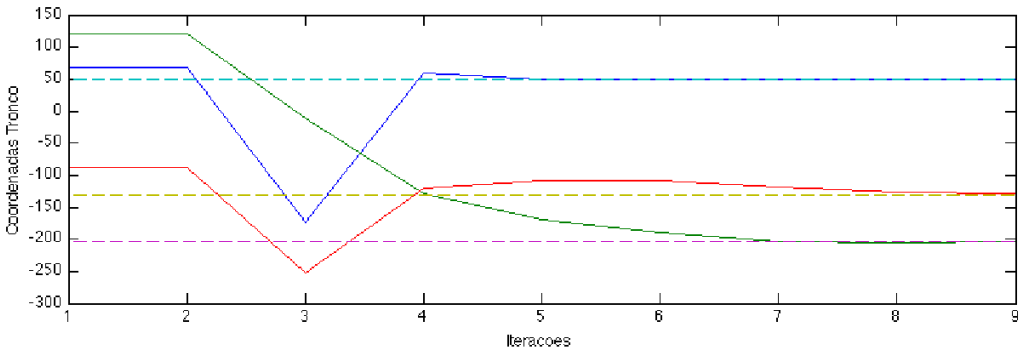

Figura 6.4: Evolução das três coordenadas do tronco em face aos valores de referência

Geração de Trajetórias das Pernas							
Passo	Acertos	Falhas	%	$\varepsilon_{min\ (mm)}$	$\varepsilon_{max\ (mm)}$	$\varepsilon_{avg\ (mm)}$	it_{avg}
a = 0.001; b = 20; c = 0.05	4306	694	86.12	0.4636	12.2529	1.4155	15.02
a = 0.001; b = 20; c = 0.005	3792	1208	75.84	0.6231	10.3606	1.5732	14.15
a = 0.001; b = 40; c = 0.05	4648	352	92.96	0.6208	7.8964	2.2246	13.66
a = 0.005; b = 20; c = 0.05	556	4444	11.12	0.6670	6.0040	1.2525	9.55
a = 0.002; b = 10; c = 0.02	4415	585	88.30	0.2791	8.3542	1.4382	16.70
a = 0.002; b = 10; c = 0.002	3852	1148	77.04	0.6112	8.3866	1.3894	15.67
a = 0.002; b = 60; c = 0.02	2474	2526	49.48	0.0000	6.0912	1.6194	7.37
a = 0.007; b = 10; c = 0.02	659	4341	13.18	0.6133	8.2018	1.3506	11.91

Tabela 6.4: Resultados do algoritmo de geração de trajectórias das pernas

um teste bem sucedido quando o algoritmo converge para uma solução apenas numa tentativa (antes de atingir o número limite de iterações admissível – 50). Recorde-se a equação 4.25, que determina o valor do passo a aplicar em cada iteração, em função do erro em relação à posição final.

$$step(e) = \begin{cases} a.e^2 + b & se\ step(e) \leq c \\ c & se\ step(e) > c \end{cases} \tag{6.1}$$

A elevada percentagem de acertos nuns casos e a baixa percentagem noutros (resultados visíveis na tabela 6.4) permite comprovar o efeito do passo variável no algoritmo proposto, e verificar a importância da escolha de parâmetros adequados, não sendo todavia possível extrair conclusões concretas sobre que valores este deve tomar como parâmetros para atingir o seu melhor desempenho. Contudo, a existência de casos de insucesso está também associada com as limitações impostas às articulações. A tabela 6.5 relaciona o número de ocorrências de insucesso com o número de ângulos que atingiram os seus limites físicos. Estes são os casos em que uma ou várias articulações apenas poderiam evoluir no sentido que as afastariam do objectivo cuja função-custo se pretende minimizar.

Relação entre as Falhas e o Número de Articulações no Respectivo Limite								
Passo	0	1	2	3	4	5	6	Total
a = 0.001; b = 20; c = 0.05	102	39	22	20	509	2	0	694
a = 0.001; b = 20; c = 0.005	176	282	122	65	561	2	0	1208
a = 0.001; b = 40; c = 0.05	23	10	0	25	285	9	0	352
a = 0.005; b = 20; c = 0.05	5	12	20	269	2085	2053	0	4444
a = 0.002; b = 10; c = 0.02	59	57	42	76	350	1	0	585
a = 0.002; b = 10; c = 0.002	109	395	80	132	431	1	0	1148
a = 0.002; b = 60; c = 0.02	1	10	121	0	783	746	865	2526
a = 0.007; b = 10; c = 0.02	280	600	1041	837	1213	370	0	4341

Tabela 6.5: Relação entre o número de falhas e o número de articulações no respectivo limite

WIIMOTE		
#	**Descrição**	**Resultado**
4.1.1	O botão ↑ inicia no robô um novo ficheiro Matlab	✔
4.1.2	O botão ↓ termina no robô o registo num ficheiro Matlab	✔
4.1.3	O botão 'A' imobiliza o robô na ultima posição recebida, impedindo o envio de novas tramas	✔
4.1.4	O botão 'B' imobiliza o robô com os braços esticados para a frente, impedindo o envio de novas tramas	✔
4.1.5	Os botões '+' e '-' incrementam e decrementam, respectivamente, o valor do coeficiente do FPB	✔
4.1.6	A combinação dos botões 'A' e 'B' termina o programa e todos os ficheiros que tenham entretanto sido criados	✔

Tabela 6.6: *Checklist* de controlo do Wiimote

No entanto, é importante ressalvar que, embora estes casos sejam matematicamente exequíveis, são fisicamente improváveis. Uma vez que o processo de marcha do robô é um processo continuado, é esperado que os ângulos iniciais não estejam já muito longe daqueles que serão os seus valores finais, para um dado ponto de destino.

6.4 Controlo do Wiimote

Testar o controlo do Wiimote passa unicamente por avaliar o efeito introduzido aos pressionar os vários botões (tabela 6.6).

6.5 Execução de Movimentos

À semelhança do algoritmo de determinação de ângulos para a geração de trajectórias, a resposta do NAO aos movimentos captados pelo Kinect constitui um dos grandes resultados deste projecto. Testar empiricamente a aplicação com o robot NAO é um método que facilmente se percebe se satisfaz os requisitos exigidos, e se a resposta do NAO corresponde ao esperado. Assim, a página *web* da Dissertação[2] contém um conjunto de vídeos que exploram várias sequências de movimentos de maior ou menor complexidade. O anexo D contém um conjunto de fotografias de um mesmo vídeo, onde é possível comparar sucessivas posições com as respectivas respostas por parte do NAO.

A figura 6.5 consiste numa comparação entre os ângulos enviados pelo computador para o NAO e os valores actuados nas articulações. Este ensaio consiste na descrição de um movimento oscilatório com o braço direito, caracterizado pela variação da primeira articulação do ombro (RShoulderPitch).

[2]http://www.fe.up.pt/~ee06161

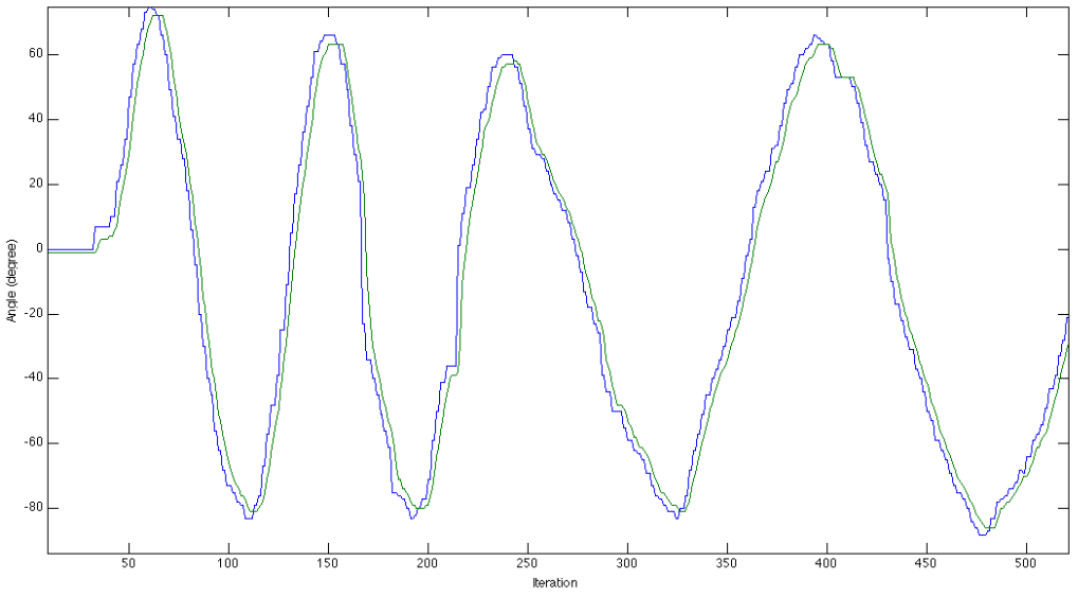

Figura 6.5: Ângulo enviado para o NAO *(a azul)* e o ângulo actuado na articulação *(a verde)*, respeitante ao ombro direito (RShoulderPitch)

6.6 Ensino por Demonstração

O último teste incide sobre os mecanismos criados de forma a possibilitar a reprodução posterior de sequências de movimentos (tabela 6.7).

6.7 Sumário

Este capítulo descreveu todos os procedimentos considerados para verificar a validade da solução proposta. A implementação superou todas as exigências identificadas, sem grandes surpresas, resultando numa aplicação robusta que permite o controlo do robô NAO à distância, com recurso a uma rede Wi-Fi e a uma câmara com visão em profundidade.

Do ensino por demonstração resultam ficheiros em formato de texto, que permitem agregar apenas a informação que é indispensável ao processo, resultando em ficheiros de tamanho reduzido mas ricos em informação. Por outro lado, os ficheiros em formato Matlab permitem facilmente apurar se o robot responde correctamente face às mensagens que recebe, e ainda conferir o efeito que o filtro passa-baixo dos actuadores impõe na sua resposta. É possível também avaliar o efeito do filtro implementado à saída do Kinect e, face à frequência que se deseja ver cortada, escolher diferentes valores do coeficiente deste.

O robô foi testado exaustivamente, tendo sido exploradas diversas combinações de ângulos, e diversos movimentos, não tendo sido encontradas quaisquer respostas que não as que teoricamente se esperavam. A programação das várias funções segundo uma arquitectura de processamento

ENSINO POR DEMONSTRAÇÃO		
#	**Descrição**	**Resultado**
6.1.1	O arranque da aplicação em modo *NewRecord* automaticamente inicia um novo ficheiro	✔
6.1.2	O ficheiro é actualizado a uma cadência pré-definida, igual à cadência com que a informação é enviada pela rede.	✔
6.1.3	A cada linha contendo a mensagem a enviar guardada no ficheiro é anexada uma referência temporal, indicando o número de repetições dessa linha.	✔
6.1.4	O arranque da aplicação em modo *UseRecord* carrega correctamente o ficheiro	✔
6.1.5	A rotina *UseRecord* separa correctamente a referência temporal da restante mensagem	✔
6.1.6	A rotina é executada de forma cíclica, respeitando os intervalos de tempo expressos na referência temporal	✔

Tabela 6.7: *Checklist* de ensino por demonstração

paralelo permitiu ainda poupar a rede a eventuais congestionamentos, resultando num fluxo de mensagens regular e continuado, não afectando o decorrer do resto da aplicação.

Capítulo 7

Conclusões e Trabalhos Futuros

Este capitulo sumariza as principais conclusões obtidas, e apresenta um conjunto de melhorias do sistema projectado. Por fim, é feito um levantamento de trabalhos futuros passíveis de ser realizados no âmbito de dissertações ou outros projectos relacionados com o trabalho desenvolvido.

7.1 Conclusão

O NAO é uma tecnologia recente, e mais recente ainda na comunidade da FEUP[1]. Tendo sido adquirido há menos de um ano, um dos principais objectivos consistia na familiarização com o robô, e com a arquitectura que o caracteriza. Desta forma, procurou-se conhecer o NAO, para mais tarde responder a uma pergunta: *Será possível e exequível controlar um robô humanóide à distância por mimetização, sem recorrer a tecnologias invasivas e sem quaisquer restrições ao seu movimento?*

O facto de anteriormente a liberdade de movimento se encontrar condicionada atrás de um *joystick* ou de um teclado de um computador tornou possível conduzir este projecto no sentido de aproximar o conhecimento científico da realidade constante da pergunta, através de uma demonstração prática robusta e totalmente funcional. De facto, a solução que é aqui apresentada responde positivamente aos desafios que a pergunta coloca. A utilização de uma câmara – o Kinect – constitui um mecanismo perfeitamente não invasivo para o ser humano, não impondo quaisquer limitações ao movimento que não aquelas intrínsecas ao robô em si.

O robô humanóide utilizado neste projecto é um de muitos já existentes. É pequeno, de baixo custo (quando comparado a outros modelos existentes), mas limitado em termos de manobrabilidade. Os seus actuadores são mais fracos que os de marcas concorrentes, e o plástico técnico que o reveste não oferece grande resistência a esforços continuados. No entanto, foi suficiente para mostrar que é possível, com equipamento de baixo custo, implementar um algoritmo de imitação, não só em simultâneo como em qualquer instante mais tarde, com um respeito rigoroso pelas restrições temporais, características do processo em controlo. Tais modos de operação, que se viram nesta dissertação serem praticáveis, podem ser aplicados numa panóplia de situações. Assim, foi

[1] Faculdade de Engenharia da Universidade do Porto

possível demonstrar as várias capacidades do NAO que foram exploradas ao longo da dissertação, em diversos eventos nos quais a FEUP e o INESC[2]-Porto participaram, tendo despertado a atenção de imensas pessoas. Permitiu-se também que algumas interagissem com o NAO, colocando-se elas à frente do Kinect, tendo o sistema funcionado como esperado, e sem qualquer problema.

7.2 Melhoramentos

O sistema conforme foi projectado e implementado mostrou provas da sua robustez. O Kinect apresenta-se no mercado como a solução de mais baixo custo actualmente capaz de devolver valores de profundidade para um plano. A utilização desta câmara no projecto acarretou no entanto a recepção de imagens sujeitas a bastante ruído (falha entretanto colmatada), resultando numa solução inicial em que o robô tremia bastante. A troca de mensagens realiza-se segundo uma ligação ponto-a-ponto, sendo negado o acesso de múltiplos dispositivos ao NAO. O ensino por demonstração guarda as mensagens em ficheiros de tamanho reduzido.

No entanto, algumas melhorias podem ser feitas. O envio de ordens para o NAO é feito a uma cadência fixa, influenciada pelo tempo de execução da rotina, e pelo efeito da chamada da função *usleep(x)*. Tornar este valor arbitrário permitiria aumentar a rapidez da resposta, embora exigindo à rede um maior fluxo de informação. Por outro lado, dotar os ficheiros associados às várias sequências de movimentos (modo *UseRecord*) de um cabeçalho que contivesse outras informações (como por exemplo a cadência de envio de mensagens pela rede, anteriormente definida) permitiria ao operador arbitrar a velocidade com que desejaria ver os seus movimentos repetidos.

Os capítulos 4 e 6 procuraram também descrever, respectivamente, um algoritmo que permitisse gerar combinações de seis ângulos que colocassem a extremidade de um manipulador numa dada posição, e os respectivos resultados. Embora tal abordagem tenha produzido soluções matematicamente válidas, muitas das restrições físicas foram ignoradas e merecem, numa eventual próxima oportunidade, uma particular atenção. Uma delas prende-se com o facto das duas articulações *LeftHipYawPitch* e *RightHipYawPitch* partilharem o mesmo motor, pelo que os valores actuados nas duas articulações deverão ser iguais. Assim, aquele que era um sistema de três equações e seis incógnitas cresce, pela dependência destas duas variáveis, num sistema de seis equações e onze incógnitas. Ainda, deve ser procurado conhecer os valores razoáveis de cada articulação, e favorecer a sua permanência nesses valores. A título exemplificativo, as articulações do tornozelo deverão ser determinadas de forma a garantir que o apoio se dará com o pé colocado de forma paralela ao chão. Por outro lado, outros parâmetros do algoritmo devem ser reavaliados. Assim, deve ser procurado conhecer o critério de geração do passo que mais favorecerá a marcha do robô. Deve ser também procurado analisar de que forma os casos em que as articulações atingem o seu limite devem ser geridos pelo algoritmo.

[2]Instituto de Engenharia de Sistemas e Computadores

7.3 Trabalhos Futuros

A aplicação desenvolvida debruçou-se sobre a manipulação do robô por interacção com os seus actuadores associados aos braços. Procurou-se também aplicar os conhecimentos adquiridos na manipulação das pernas, apesar de limitados a um ambiente de simulação. Embora completar este algoritmo não fosse objectivo desta dissertação, desenvolver a totalidade do sistema de locomoção em ambiente de simulação permitirá fazer uma análise comparativa entre este e outros métodos de locomoção actualmente existentes (alguns implementados pelas várias equipas de futebol robótico), e averiguar as fraquezas e valências dos vários algoritmos. Por outro lado será um desafio interessante poder explorar novos mecanismos de teleoperação associados à locomoção do robô, e investigar de que forma o Kinect e/ou o Wiimote (ou outros controladores) poderão ser utilizados de modo a garantir um controlo de fácil execução, de alto desempenho, e de menor consumo energético.

O robô NAO é demasiado complexo para que em alguns meses se possam explorar todas as suas funcionalidades. Por isso, vários actuadores e sensores do NAO ficaram por abordar, mas cuja compreensão e utilização aumentam grandemente o seu desempenho. A indisponibilidade do robô durante parte do projecto inviabilizou um estudo exaustivo do seu sistema de visão e de outros dispositivos de sensorização do meio envolvente (como os ultra-sons, microfones e infra-vermelhos), muito importantes para garantir um grau de autonomia elevado, e imprescindíveis nas competições de futebol robótico nas quais a FEUP está envolvida.

Fica também presente um outro desafio: o de alargar os conceitos adquiridos sobre o NAO a mais do que um robô, e dotá-los da capacidade de comunicarem e partilharem informações sobre o meio que os envolve, com vista à realização de tarefas em ambiente de cooperação.

Anexo A

Principais Funções Implementadas

Este anexo procura documentar as principais funções implementadas ao longo das aplicações, tanto a que executa no computador como a que executa no robô NAO. Para cada função são indicados os dados de entrada, de saída, e uma descrição sumária da mesma.

A.1 Aplicação Local

A.1.1 *network.h*

A biblioteca *network.h* procurou agregar todas as funções respeitantes à instanciação e gestão de ligações entre dispositivos.

Função	**Argumentos**	**Resultado**	**Descrição**
constructServer()	Endereço IP Porta	Servidor	Esta função configura um novo cliente numa nova ligação ponto-a-ponto, com outra máquina identificada pelo IP, cuja aplicação é definida pela porta.
constructServer()	Endereço IP	Servidor	Esta função configura um novo servidor, capaz de receber múltiplas ligações, atribuindo à aplicação uma porta predefinida.
openSocketConnection()	socket Servidor	N/A	Associa a configuração de uma ligação do tipo "cliente"a uma *socket*.
sendMessage()	socket Mensagem	msglength	Envia uma mensagem pela rede TCP/IP, para o destinatário identificado pela *socket*, devolvendo o número de caracteres enviados.
receiveMessage()	socket buffer	N/A	Armazena num espaço de memória indicado pelo *buffer* uma mensagem recebida através da rede TCP/IP, de um remetente identificado pela *socket*.

bindServerSocket()	socket Servidor	N/A	Associa a configuração de uma ligação do tipo "servidor"a uma *socket*.
listenToServer()	socket	N/A	Processa a recepção de novas mensagens através da *socket* do tipo "servidor".
newClientConnection	socket	Cliente	Aguarda a ligação de um novo cliente ao servidor associado à *socket*, devolvendo em caso de sucesso uma estrutura contendo as informações do cliente.

A.1.2 *wiimote.h*

A biblioteca *wiimote.h* procurou agregar todas as funções respeitantes à gestão de informação proveniente do comando Wiimote.

Função	**Argumentos**	**Resultado**	**Descrição**
HandleEvent()	WiiMote	Botoes[12]	Quando um botão é pressionado ou largado, esta função analisa a informação proveniente do comando, organizando-a num vector que agrega a informação actualizada de cada botão

A.1.3 *nao.h*

A biblioteca *nao.h* procurou agregar todas as funções respeitantes à instanciação e gestão das principais classes implementadas, bem como todos os mecanismos de análise da cinemática do robô.

Função	**Argumentos**	**Resultado**	**Descrição**
invKinematics()	NaoJoint[8] NaoPoint[13]	N/A	Agrega a invocação de todas as funções associadas ao processamento da cinemática inversa. NaoPoint[13] é um vector que contém as coordenadas dos pontos das articulações do esqueleto humano. NaoJoint[8] é um vector onde serão armazenados os valores a dos ângulos a aplicar a cada uma das articulações do NAO, numa gama de 0º a 360º, com uma resolução de 0.1º.

kineLeftArm()	NaoJoint[8] Ombro Esq. Cotovelo Esq. Mão Esq.	N/A	Com base nas coordenadas das articulações do braço esquerdo, determina os ângulos a aplicar às articulações associadas a este. O resultado é armazenado nas quatro primeiras posições do vector NaoJoint[8].
kineRightArm()	NaoJoint[8] Ombro Dir. Cotovelo Dir. Mão Dir.	N/A	Com base nas coordenadas das articulações do braço direito, determina os ângulos a aplicar às articulações associadas a este. O resultado é armazenado nas quatro segundas posições do vector NaoJoint[8].
armNormalization()	theta_rad limite_inf limite_sup	theta_deg	Recebido um ângulo em radianos, devolve o respectivo ângulo em graus, desde que dentro de uma gama predefinida.
getCoords()	theta_1 theta_2 NaoPoint A	NaoPoint B	Converte e devolve as coordenadas de um ponto expresso num referencial A num referencia B, quando os dois estão relacionados por uma rotação de theta_1 graus segundo o eixo e_y e theta_2 graus segundo o eixo e_x, no referencial A.
NaoPointsInit()	NaoPoint[13]	N/A	Inicializa cada uma das posições de um vector de pontos (articulações do corpo).
updateJointCoords()	NaoPoint[13]	N/A	Descreve o procedimento que analisa a informação proveniente da câmara Kinect, e invoca as rotinas associadas à classe NaoPoint que permitem actualizar a informação relevante ao resto do processo.

Uma vez que a linguagem C++ contempla a possibilidade de criar classes, agregando variáveis e funções respeitantes a uma dada entidade, foi criada a classe *NaoPoint* cujas várias instâncias se encontram associadas a cada uma das articulações do esqueleto humano. Assim, esta classe contém três variáveis públicas, *x*, *y* e *z* (do tipo *double*), e ainda 4 variáveis privadas, dos tipos *XnSkeletonJointPosition*, *XnPoint3D* e *XnSkeletonJoint*, constantes das rotinas das bibliotecas da *OpenNI*, que permitem a interface entre o Kinect e o computador. Também, as seguintes funções foram implementadas:

Função	**Argumentos**	**Resultado**	**Descrição**

getCoords()	User ID	x y z	Actualiza as coordenadas *x*, *y* e *z* do *NaoPoint* associado, com as informações extraídas de um utilizado indicado aquando da sua invocação
setJoint()	N/A	N/A	Define a que articulação está associado o *NaoPoint* instanciado

A.1.4 *main.cpp*

main.cpp consiste no principal ficheiro da aplicação, que invoca todas as outras bibliotecas, e executa as funções associadas à execução da aplicação.

Função	**Argumentos**	**Resultado**	**Descrição**
NewRecordFunc()	N/A	N/A	Função que contém as rotinas associadas ao processamento do programa no modo *NewRecord*, invocando funções de processamento de análise cinemática, abertura, escrita e fecho de um ficheiro que contenha a rotina programada, e de comunicação com o robô NAO.
UseRecordFunc()	N/A	N/A	Função que contém as rotinas associadas ao processamento do programa no modo *UseRecord*, invocando funções de abertura, leitura e fecho de um ficheiro que contenha a rotina programada, e de comunicação com o robô NAO.
LiveFunc()	N/A	N/A	Função que contém as rotinas associadas ao processamento do programa no modo *Live*, invocando funções de processamento de análise cinemática, e de comunicação com o robô NAO. Esta função contém aínda um algoritmo que permite manter um registo dos ângulos enviados para o robô, utilizado para eventuais rastreios de erros.

poseDetect_tf()	N/A	N/A	Função executada paralelamente ao resto do processo. De cada vez que uma nova pessoa se calibra perante o sistema, esta função invoca uma das duas funções (*NewRecordFunc()* ou *LiveFunc()*, dependendo dos argumentos indicados aquando da iniciação da aplicação). Uma vez terminadas essas funções (quer pela perda da pessoa, quer pelo accionamento de uma combinação de botões do comando Wiimote configurada para terminar a execução de uma das duas funções acima indicadas, esta função envia ainda uma última mensagem ao robô NAO – "0000000000000000000000000000000--de forma a garantir que este, em repouso, se encontra numa posição que não ponha em causa a sua integridade.

A.2 Aplicação Remota

Função	**Argumentos**	**Resultado**	**Descrição**
naoJointsInit()	NaoJoint[8]	N/A	Atribui a cada elemento do vector um atributo que identifica a qual das oito articulações dos braços corresponde.
commitNaoJoints()	NaoJoint[8]	N/A	Executa a rotina que associa um apontador ao primeiro registo da memória partilhada, invocando de seguida a função *printJoints()*.
printJoints()	NaoJoint[8]	N/A	Escreve na memória partilhada o valor a aplicar a cada uma das articulações.

updateCoords()	NaoJoint[8]	N/A	Converte a informação mais actual dos ângulos recebidos pela rede de graus para radianos, convertendo para o seu simétrico quando necessário. O resultado desta função consiste num vector que contém os ângulos a aplicar nas articulações, depois filtrados por um filtro passa-baixo (caso o seu coeficiente seja diferente de zero). É também nesta função que um ficheiro é preenchido (legível em Matlab), contendo um registo completo dos ângulos recebidos pela rede – referências – e os ângulos reais das articulações.
networkComm_tf()	N/A	N/A	Função que executa paralelamente ao resto do processo. Esta função mantém um servidor que aceita apenas uma ligação de um cliente, via TCP/IP.
getAngles()	Mensagem	int[8]	Avalia o conteúdo da mensagem de 32 caracteres, guardando num vector de oito inteiros um número que, numa escala de 0000 a 3600, representa uma referência para cada um dos oito ângulos das articulações do braço, com uma resolução de 0.1º.
handleClient()	socket Mensagem	Ângulos[8] Coef. FPB[1]	Aquando da recepção de uma mensagem, avalia o valor do primeiro caracter. Consoante o resultado, um de três algoritmos é executado. Quando o valor é '$', o coeficiente do filtro passa-baixo dos actuadores é reconfigurado para valor indicado pelos três dígitos subsequentes (décimas, centésimas e milésimas, respectivamente); quando é '#'", o programa inicia um ficheiro vazio onde serão registados os valores dos ângulos recebidos pela rede e os ângulos reais das articulações, ou o ficheiro actual é encerrado para posterior descarregamento (consoante o valor do segundo caracter seja '0' ou '2'). Restantes valores para o primeiro caracter implicam a chamada da função *getAngles()*.

[1] Filtro Passa-Baixo

getNewAngle()	Coef. FPB oldAng refAng	newAng	Gera um novo ângulo para uma data articulação, após a passagem por um filtro passa-baixo.

Anexo B

Código-fonte Matlab

B.1 Cálculo do Jacobiano dos Braços

```
1  syms C1 S1 C2 S2 C3 S3 C4 S4 x d e
2
3  H0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
4  H1 = [C1 0 S1 0; S1 0 -C1 0; 0 1 0 0; 0 0 0 1];
5  H2 = [-S2 0 C2 d*C2; C2 0 S2 d*S2; 0 1 0 0; 0 0 0 1];
6  H3 = [C3 0 -S3 0; S3 0 C3 0; 0 -1 0 0; 0 0 0 1];
7  H4 = [C4 -S4 0 e*S4; S4 C4 0 -e*C4; 0 0 1 0; 0 0 0 1];
8
9  W0 = H0;
10 W1 = H0*H1;
11 W2 = H0*H1*H2;
12 W3 = H0*H1*H2*H3;
13 W4 = H0*H1*H2*H3*H4;
14
15 Z0 = W0(1:3,3);
16 Z1 = W1(1:3,3);
17 Z2 = W2(1:3,3);
18 Z3 = W3(1:3,3);
19 Z4 = W4(1:3,4);
20
21 O0 = W0(1:3,4);
22 O1 = W1(1:3,4);
23 O2 = W2(1:3,4);
24 O3 = W3(1:3,4);
25 O4 = W4(1:3,4);
26
27 J = [extprod(Z0, (O4-O0)) extprod(Z1, (O4-O1)) extprod(Z2, (O4-O2)) ...
       extprod(Z3, (O4-O3));
28       Z0 Z1 Z2 Z3];
29
30 J = simplify(J);
```

```
J = trigREM(J);
J2 = J*J';
disp('conjREM'); J2 = conjREM(J2);
disp('trigREM'); J2 = trigREM(J2);
disp('With S4 = 0'); J2 = subs(J2,S4,0);
disp('With C4 = 1'); J2 = subs(J2,C4,1);
disp('Determinant'); D = det(J2);
disp('absREM'); D = absREM(D);
disp('cosREM'); D = cosREM(D);

D

J2 = J*J';
disp('conjREM'); J2 = conjREM(J2);
disp('trigREM'); J2 = trigREM(J2);
disp('With C2 = 0'); J2 = subs(J2,C2,0);
disp('With S2 = 1'); J2 = subs(J2,S2,1);
disp('Determinant'); D = det(J2);
disp('absREM'); D = absREM(D);
disp('cosREM'); D = cosREM(D);

D

% As funcoes conjREM(), trigREM(), absREM() e cosREM()
% visam simplificar os termos simbolicos, tornando
% os resultados mais legiveis
```

B.2 Análise Cinemática das Pernas

```
syms C0 C1 C2 C3 C4 C5 C6 S0 S1 S2 S3 S4 S5 S6;
syms A B C D E F G H;
a = A; b = B; c = 0; d = 0; e = E; f = F; g = 0; h = H;

M1  = [0 1 0 0; -S0 0 C0 -b; C0 0 S0 -a; 0 0 0 1];
M2a = [C1 0 -S1 0; S1 0 C1 0; 0 -1 0 0; 0 0 0 1];
M2b = [C0 S0 0 0;-S0 C0 0 0; 0 0 1 c; 0 0 0 1];
M3  = [C2 0 S2 0; S2 0 -C2 0; 0 1 0 d; 0 0 0 1];
M4  = [C3 -S3 0 -e*S3; S3 C3 0 e*C3; 0 0 1 0; 0 0 0 1];
M5  = [C4 -S4 0 -f*S4; S4 C4 0 f*C4; 0 0 1 0; 0 0 0 1];
M6  = [C5 0 -S5 -g*S5; S5 0 C5 g*C5; 0 -1 0 0; 0 0 0 1];
M7  = [S6 0 -C6 -h*C6; -C6 0 -S6 -h*S6; 0 1 0 0; 0 0 0 1];

M2 = M2a*M2b;

H  = M1*M2*M3*M4*M5*M6*M7;
H_ = inv(M7)*inv(M6)*inv(M5)*inv(M4)*inv(M3)*inv(M2)*inv(M1);
```

Anexo C

Características do Robô NAO

O presente anexo, retirado da documentação da Aldebaran Robotics (41; 55) especifica as principais características do robô NAO, invocadas ao longo desta Dissertação.

C.1 Especificações Técnicas

- **Peso: 4.3 Kg**
- **Altura: 58 cm**
- **Graus de Liberdade**
  - Cabeça: 2
  - Braços: 5 em cada braço
  - Bacia: 1
  - Pernas: 5 em cada perna
  - Mãos: 2 em cada mão (0 na versão Robocup)
- **Sensores**
  - 32 x Sensores de Efeito de Hall
  - 1 x Giroscópio de 2 eixos
  - 1 x Acelerómetro de 3 eixos
  - 2 x Bumpers (à frente de cada pé)
  - 2 x Pares emissor-receptor de ultra-sons (40kHz, alcançe [0.2 a 1.2] ± 0.009 metros, com cone de abertura 60º)
  - 2 x Infra-vermelhos
  - 2 x Câmaras (VGA 640x640, 30 fps)
- **Motherboard**

  - x86 AMD Geode 500Mhz CPU
  - Memória SDRAM 256Mb / 1 Gb Memória Flash

- ***Sofware***
  - Sistema operativo: Embedded Linux (32 bit x86 ELF)
  - Linguagens de programação: C, C++, Python, Urbi

C.2 Limites das Articulações

Tabela C.1: Limites das Articulações do NAO

Articulação	**Mínimo (º)**		**Máximo (º)**	
	Left	**Right**	**Left**	**Right**
HeadYaw	-119.5		119.5	
HeadPitch	-38.5		29.5	
ShoulderPitch	-119.5	-119.5	119.5	119.5
ShoulderRoll	0.5	-94.5	94.5	-0.5
ElbowYaw	-119.5	-119.5	119.5	119.5
ElbowRow	-89.5	0.5	-0.5	89.5
WristYaw	-104.5	-104.5	104.5	104.5
HipYawPitch[1]	-65.62		42.44	
Hipoll	-21.74	-42.30	45.29	23.76
HipPitch	-101.63	-101.54	27.73	27.82
KneePitch	-5.29	-5.90	121.04	121.47
AnklePitch	-68.15	-67.97	52.85	53.40
Ankleoll	-44.06	-22.27	22.79	45.03

[1] As pernas esquerda e direita partilham o mesmo motor, pelo que as duas articulações não podem ser controladas de forma indepedente

Anexo D

A Aplicação em Funcionamento

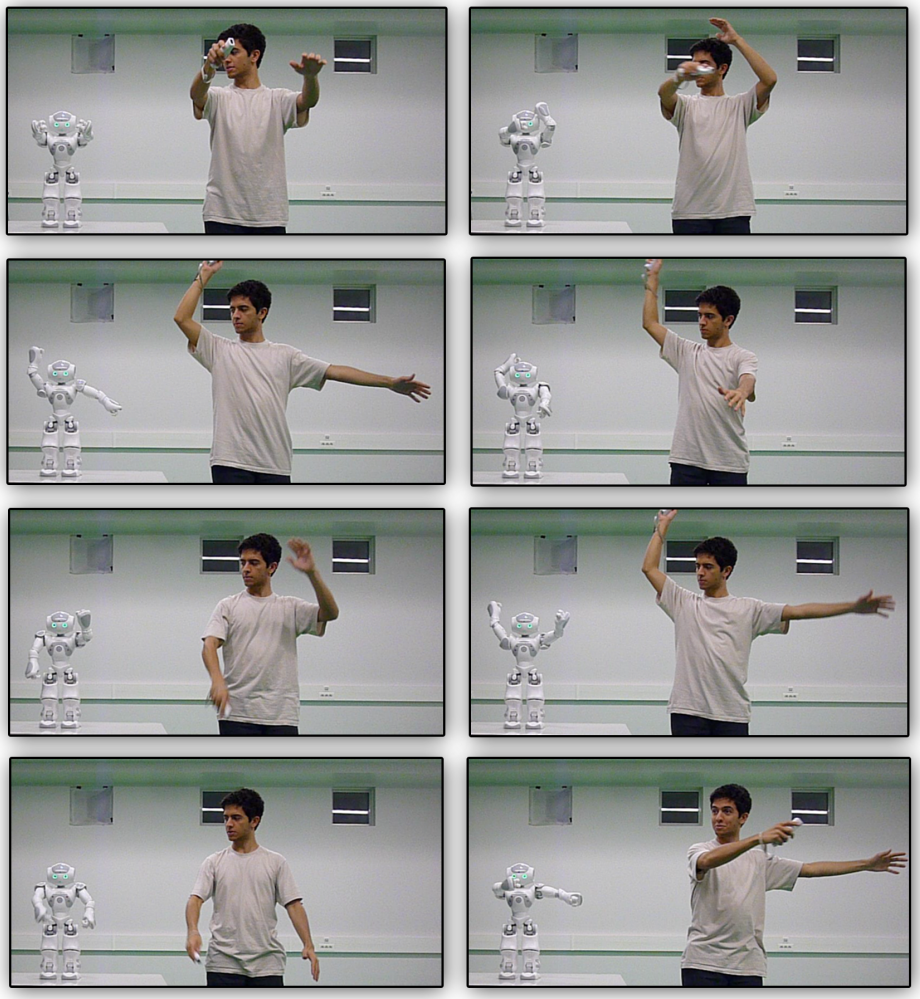

Figura D.1: A aplicação na prática

Referências

[1] NASA. A short history of robots, March. 2003. http://prime.jsc.nasa.gov/ROV/history.html.

[2] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, e Bruno Maisonnier. Mechatronic design of nao humanoid. Em *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, páginas 769 –774, Maio 2009.

[3] D. Gouaillier e P. Blazevic. A mechatronic platform, the aldebaran robotics humanoid robot. Em *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, páginas 4049 –4053, 2006.

[4] Aldebaran Robotics. *NAO Datasheet*, 2005.

[5] G. Richards. It shoots - it scores! [control robotics]. *Engineering Technology*, 5(8):38 –40, 2010.

[6] The Robocup Federation. Robocup official website, 2011. http://www.robocup.org/.

[7] Aldebaran Robotics. Aldebaran robotics webpage, 2011. http://www.aldebaran-robotics.com.

[8] Robocup Technical Committee. Robocup standart platform league (nao) rule book, 2008. http://www.tzi.de/spl/pub/Website/Downloads/NaoRules2008.pdf.

[9] N. Shafii. Machine learning and optimization applied to bipedal locomotion of simulated and real humanoid robots. july 2010.

[10] Asimo the robot: A world-renowned robot walks the world. http://www.suite101.com/content/asimo-the-robot-a53854.

[11] Wikipedia (fr) : Zero_moment_point. http://fr.wikipedia.org/wiki/Zero_Moment_Point.

[12] K. Erbatur e U. Seven. An inverted pendulum based approach to biped trajectory generation with swing leg dynamics. Em *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, páginas 216 –221, 2007.

[13] D. Gouaillier, C. Collette, e C. Kilner. Omni-directional closed-loop walk for nao. Em *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, páginas 448 –454, dec. 2010.

[14] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, e H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. Em *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, páginas 1620 – 1626 vol.2, sept. 2003.

[15] J. Strom, G. Slavov, e E. Chown. Omni-directional walking using zmp and preview control for the nao humanoid robot.

[16] Wikipedia (en) : Spline_(mathematics). `http://en.wikipedia.org/wiki/Spline_(mathematics)`.

[17] J. M. Lima. *Construção de um Modelo Realista e Controlo de um Robô Humanóide*. Tese de doutoramento, Faculdade de Engenharia da Universidade do Porto, 2009.

[18] K. Inoue, T. Sumi, e Shugen Ma. Cpg-based control of a simulated snake-like robot adaptable to changing ground friction. Em *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, páginas 1957 –1962, 2007.

[19] A. Pina Martins e R. Araújo. Neural networks for control. Faculdade de Engenharia da Universidade do Porto, 2010.

[20] Wikipedia (en) : Fourier_series. `http://en.wikipedia.org/wiki/Fourier_series`.

[21] L. Yang, C.M. Chew, A.N. Poo, e T. Zielinska. Adjustable bipedal gait generation using genetic algorithm optimized fourier series formulation. Em *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, páginas 4435 –4440, 2006.

[22] N. Shafii, A. Khorsandian, A. Abdolmaleki, e B. Jozi. An optimized gait generator based on fourier series towards fast and robust biped locomotion involving arms swing. Em *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on*, páginas 2018 –2023, 2009.

[23] N. Shafii, L.P. Reis, e N. Lau. Biped walking using coronal and sagittal movements based on truncated fourier series. Em *Robocup-2009: Robot Soccer World Cup XIII*, 2010.

[24] Wikipedia (en) : Autonomous_robot. `http://en.wikipedia.org/wiki/Autonomous_robot`.

[25] J. Cardoso e A.M. Mendonça. Sistemas baseados em visão. Faculdade de Engenharia da Universidade do Porto, 2009.

[26] R. Thomas. *Applied Image Processing*. McGraw-Hill International Edition, 1996.

[27] A.P. Moreira. Sistemas robóticos autónomos. Faculdade de Engenharia da Universidade do Porto.

[28] L. G. Shapiro e G. C. Stockman. *Computer Vision*. Prentice-Hall, 2001.

[29] M. Pena, I. Lopez, e R. Osorio. Invariant object recognition robot vision system for assembly. Em *Electronics, Robotics and Automotive Mechanics Conference, 2006*, volume 1, páginas 30 –36, 2006.

[30] Yang Xudong, Dai Peng, He Ping, e Li Pan. Intelligent detection of convex polygon based on hough transformation and set classifier. Em *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, páginas 1 –4, Maio 2010.

[31] S. Tsuji e F. Matsumoto. Detection of ellipses by a modified hough transformation. *Computers, IEEE Transactions on*, C-27(8):777 –781, 1978.

[32] Prajin Palungsuntikul e Wichian Premchaiswadi. Object detection and keep on a mobile robot by using a low cost embedded color vision system. Em *Knowledge Engineering, 2010 8th International Conference on ICT and*, páginas 70 –76, 2010.

[33] Black Ice Software. Hsi color conversion - imaging toolkit feature. `http://www.blackice.com/colorspaceHSI.htm`.

[34] Thomas Röfer, Tim Laue, Judith Müller, Armin Burchardt, Erik Damrose, Alexander Fabisch, Fynn Feldpausch, Katharina Gillmann, Colin Graf, Thijs Jeffry de Haas, Alexander Härtl, Daniel Honsel, Philipp Kastner, Tobias Kastner, Benjamin Markowsky, Michael Mester, Jonas Peter, Ole Jan Lars Riemann, Martin Ring, Wiebke Sauerland, André Schreck, Ingo Sieverdingbeck, Felix Wenk, e Jan-Hendrik Worch. B-human team report and code release 2010, 2010. Only available online: `http://www.b-human.de/file_download/33/bhuman10_coderelease.pdf`.

[35] Wikipedia (en) : Kinect. `http://en.wikipedia.org/wiki/Kinect`.

[36] iFixit. Microsoft kinect teardown, 2011. `http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1`.

[37] Open kinect project, feb. 2011. `http://www.openkinect.org`.

[38] K. Konolige e P. Mihelich. Technical description of kinect calibration, dec. 2010. `http://www.ros.org/wiki/kinect_calibration/technical`.

[39] M. W. Spong, S. Hutchinson, e M. Vidyasagar. *Robot Modeling and Control*. John Wiley Sons, Inc.

[40] A.P. Moreira. Robótica industrial. Faculdade de Engenharia da Universidade do Porto, 2010.

[41] T. G. Sanchez. Artificial vision in the nao humanoid robot. Tese de mestrado, Universitat Rovira I Virgili, sept. 2009.

[42] R.L. Burden e J.D. Faires. *Numerical analysis*. Prindle, Weber & Schmidt series in mathematics. PWS-KENT Pub. Co., 1989.

[43] Sooyong Lee, Dae-Seong Choi, Munsang Kim, Chong-Won Lee, e Jae-Bok Song. Human and robot integrated teleoperation. Em *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 2, páginas 1213 –1218 vol.2, oct 1998.

[44] N.E. Sian, K. Yokoi, S. Kajita, F. Kanehiro, e K. Tanie. Whole body teleoperation of a humanoid robot - development of a simple master device using joysticks. Em *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, páginas 2569 – 2574 vol.3, 2002.

[45] T. Takubo, K. Inoue, T. Arai, e K. Nishii. Wholebody teleoperation for humanoid robot by marionette system. Em *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, páginas 4459 –4465, oct. 2006.

[46] Lei Zhang, Qiang Huang, Qiusheng Liu, Tao Liu, Dongguang Li, e Yuepin Lu. A teleoperation system for a humanoid robot with multiple information feedback and operational modes. Em *Robotics and Biomimetics (ROBIO). 2005 IEEE International Conference on*, páginas 290 –294, 0-0 2005.

[47] Yuepin Lu, Qiang Huang, Min Li, Xiaoyu Jiang, e M. Keerio. A friendly and human-based teleoperation system for humanoid robot using joystick. Em *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, páginas 2283 –2288, june 2008.

[48] Yuepin Lu, Li Liu, Shuxin Chen, e Qiang Huang. Voice based control for humanoid teleoperation. Em *Intelligent System Design and Engineering Application (ISDEA), 2010 International Conference on*, volume 2, páginas 814 –818, oct. 2010.

[49] Shandong Wu e Yimin Chen. Remote robot control using intelligent hand-held devices. Em *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, páginas 587 – 592, 2004.

[50] M. de Sousa. Sistemas embutidos e de tempo real. Faculdade de Engenharia da Universidade do Porto, 2009.

[51] A.V. Oppenheim, A.S. Willsky, e S.H. Nawab. *Signals Systems*. Upper Saddle River : Prentice Hall International, 2 edição, 1997.

[52] P. L. Santos. Processos estocásticos e filtro de kalman. Faculdade de Engenharia da Universidade do Porto, nov. 1997.

[53] F. Haugen. Derivation of a discrete-time lowpass filter. 2008.

[54] P. Portugal e L.D. Almeida. Comunicações industriais. Faculdade de Engenharia da Universidade do Porto, 2009.

[55] Aldebaran Robotics. *NAO User Guide*, v.1.6.13 edição, 2005.

[56] Wiimote - wiibrew, March. 2011. `http://wiibrew.org/wiki/Wiimote`.