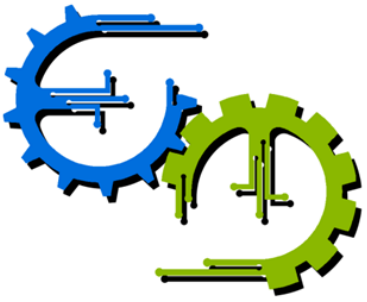
MÉTODOS DE COMUNICAÇÃO VISUAL E CONTROLE COOPERATIVO ENTRE ROBÔS HUMANOIDES

CRISTIANA MIRANDA DE FARIAS

YURI GONÇALVES ROCHA

TRABALHO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MÉTODOS DE COMUNICAÇÃO VISUAL E CONTROLE COOPERATIVO ENTRE ROBÔS HUMANOIDES

CRISTIANA MIRANDA DE FARIAS

YURI GONÇALVES ROCHA

TRABALHO DE GRADUAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO MECATRÔNICO.

APROVADA POR:

_____

Prof. Geovany Borges, ENE/UnB
(Orientador)

_____

Prof. Mariana Costa Bernardes Matias, FGA/UnB
Co-Orientadora

_____

Prof. João Ishihara, ENE/UnB
Examinador

_____

Dr. Luis Felipe da Cruz Figueredo, ENE/UnB
Examinador

BRASÍLIA, 08 DE DEZEMBRO DE 2016.

FICHA CATALOGRÁFICA

CRISTIANA MIRANDA DE FARIAS E YURI GONÇALVES ROCHA
Métodos de Comunicação Visual e Controle Cooperativo Entre Robôs Humanoides [Distrito Federal] 2016.
ix, 109p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Mecatrônica, 2016).
Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Robótica
2. Espaço Cooperativo Dual
3. Quatérnios Duais
4. Comunicação Visual

I. ENE/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

(2016). Métodos de Comunicação Visual e Controle Cooperativo Entre Robôs Humanoides, Trabalho de Graduação em Engenharia Mecatrônica, Publicação TG 23/2016, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 109p.

CESSÃO DE DIREITOS

Cristiana Miranda de Farias
Departamento de Eng. Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Yuri Gonçalves Rocha

Dedicatórias

Dedico este trabalho à minha sobrinha Júlia: que você descubra todas as maravilhas desse mundo

Cristiana Miranda de Farias

Dedico este trabalho aos meus irmãos, Igor e Sara.

Yuri Gonçalves Rocha

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a minha família, principalmente aos meus pais, Tereza Cristina e Oswaldo que sempre apoiaram cada passo que eu dei nessa minha jornada. Agradeço, obviamente, a minha avó, Lu, que sempre me acolhe com muito amor, carinho e meu prato preferido.

Gostaria de agradecer também as minhas amigas, Re, Jé, Kaká, Nati, Let, Mari, Iana, Nina e Cami. Crescer junto de vocês foi a melhor experiência do mundo. Não sei o que eu faria sem esse bonde.

Agradeço a Vigésima Sétima turma de Engenharia Mecatrônica da Universidade de Brasília. Não há dúvida que nós somos a melhor turma de todos os tempos. Em especial gostaria de agradecer aos amigos que tem estado ao meu lado desde dia um: Pedro, Eric, Daniel, De Hong, Fujita, Junior, Marco, Rodrigo, Caixeta, Jessé, Marcos.

Agradedeço também as Brighton Girls: Lu, Bá, Isa, Lê, Ana e Laís. Viver um ano na terra da rainha sem nossos Häagen-Dazs, Jantares, Viagens e Aventuras em templos obscuros seria uma afronta ao mundo.

Obrigada também a UnBeatables, melhor equipe de todas, que tem sido uma parte muito importante da minha vida nos últimos três anos. Quem eu seria sem todas as experiências que eu tive mundo afora com vocês? Obrigada Felipe, Jhon, Raphael, Ana, Gabriel, Dani, Brunovo, Bruvelho, Henrique e Eric.

Obviamente não tem como eu deixar de agradecer aos meus amigos do LARA: Rato, Roberto, Carol, Vinícius, Arthur, Miguel, Levy e Allan. Sem nossas movie nights, piscinas de bolinhas e zoeiras diversificadas eu não teria sobrevivido este ano. Obrigada por transformar um ambiente de trabalho em um lugar tão divertido. Também não posso deixar de agradecer solenemente aos meus súditos em Lara Island. Que nossa boia nunca fure.

Gostaria de deixar um agradecimento especial ao Luís. Não teríamos chegado onde estamos sem sua ajuda. Muito obrigada por estar sempre do nosso lado, com um sorriso, pronto pra tirar uma dúvida, tomar um café ou dar uma risada.

Agradeço a nossa orientadora Mariana por acreditar no nosso potencial e estar sempre ao nosso lado, ajudando a resolver qualquer problema. O Lucas será um menino de sorte, com uma mãe como você! Não poderia deixar de agradecer também ao professor Geovany, que aceitou ser nosso orientador e nos forneceu toda a estrutura necessária para este trabalho.

Finalmente, não posso deixar de agradecer ao melhor parceiro de TG do universo, Yuri, você sobreviveu! Obrigada, por toda a amizade e companheirismo nesse último ano, por ouvir minhas crises e estar sempre me apoiando. Não escolheria outra pessoa no mundo pra passar minhas noites de sábado no LARA e Frans. Juntos enfrentamos os desafios que apareceram com muito mais força! Tenho certeza do seu sucesso, amigo.

Cristiana Miranda de Farias

Agradeço primeiramente aos meus pais, Dinalva e Francisco, e aos meus irmãos, Igor e Sara, por sempre me apoiarem, me dando forças para seguir em frente e proporcionando um ambiente familiar cheio de amor e amizade.

Agradeço também aos meus amigos, Jhonantans, Henrique, Raphael, Victor, Bruno e Vilmar, que me acompanharam durante toda esta jornada na UnB. Muito obrigado por todos os momentos compartilhados. Obrigado também aos meus amigos que me acompanharam no meu intercâmbio: Altair, Filipe, Paulinha, Marcela, Mona, Gui, Nicole, Carol, Jacy. Vocês foram como minha família por um ano, e espero poder encontrá-los novamente em breve. Além disso, gostaria de agradecer aos meus amigos, Geordana, André, WooJung, JungAh, MinGyeong, Jessé e todos aqueles que me acompanharam ao longo deste ano, pela amizade e companherismo.

Muito obrigado aos meus companheiros na equipe UnBeatables: DeHong, Rafael, Dalosto, Cortes, Eric, Bruno, Brunovo, Ana, Gabriel, Eric Vale e Dani. Obrigado por todo o aprendizado compartilhado e pelas aventuras ao redor do mundo.

Agradeço também aos meus companheiros do LARA: Rato, Roberto, Vinícius, Arthur, Miguel, Marcos, Carol, Pedro, Levy, Fujita e Allan. Vocês sempre me proporcionaram um ambiente de trabalho descontraído, o que fez toda a diferença nos momentos mais difíceis decorrentes da realização deste trabalho.

Um agradecimento especial ao Luís por sempre nos auxiliar, com qualquer problema, a qualquer hora, em qualquer dia. Pessoas como você são raras hoje em dia. Meu muito obrigado pela amizade, pelo apoio e por todas as conversas aleatórias e divertidas que tivemos ao longo deste ano.

Agradeço também à professora Mariana pela orientação e por não medir esforços de modo a garantir tudo que fosse necessário para a execução de qualquer projeto. Não poderia pedir orientadora melhor. Obrigado também ao professor Geovany por fornecer todo apoio necessário para o correto funcionamento do laboratório e por aceitar ser nosso orientador.

Finalmente, gostaria de agradecer à melhor parceira de TG. Cris, posso dizer que nós formamos a dupla perfeita, e apesar de todas as noites e finais de semana no LARA e das constantes ameaças à minha integridade física, sempre nos motivamos a seguir em frente, servindo de apoio um ao outro e, principalmente, nos tornamos verdadeiros amigos.

Yuri Gonçalves Rocha

RESUMO

Este projeto busca realizar um estudo de áreas da robótica, controle e visão computacional aplicadas a robótica cooperativa. Primeiramente é proposta, levando em conta o desafio da RoboCup 2016, a implementação de um framework para comunicação de robôs a partir de técnicas de visão computacional, tirando, portanto, a necessidade de se criar uma arquitetura para redes. Em uma segunda parte, este projeto se expande em métodos de controle cooperativo entre robôs, principalmente utilizando a matemática dos quatérnios duais para a implementação do espaço cooperativo dual. Desse modo, utilizando-se a plataforma NAO, que se caracteriza como uma plataforma subatuada, foram exploradas técnicas de controle, utilizando o espaço nulo, espaço cooperativo, diferentes tipos de Jacobianas e métodos para se evitar singularidades e limites de juntas. Finalmente, o projeto se completou por uma validação física e em simulação tanto dos métodos de cooperação quanto dos métodos de comunicação visual.

ABSTRACT

This project aims to study different areas of robotics, such as the kinematic representation of manipulators, control theory and computer vision all applied to cooperative robotics. The first goal of this work is, taking into account a scenario proposed for the technical challenge of the 2016 RoboCup, to create a framework for the communication of two robots without the use of wireless networks. Aiming to expand on the idea that robots may not only communicate, but also cooperate on many different tasks, this project explores methods for control and cooperation based on movement description given by unit dual quaternions. In this way, by making use of the humanoid platform, NAO, with its underactuated manipulators, many control techniques were studied, such as the null space optimization, dual cooperative task space, different Jacobians and methods for singularity and joint limits avoidance. Finally, this project was completed by a physical and simulated validation of both the cooperation and communication methods.

Lista de Figuras

Lista de Tabelas

$\mathbb{R}$	Conjunto dos números reais
$\mathbb{R}^n$	Sequência de n números reais em forma de vetores ou matrizes
$\mathbb{C}$	Conjunto dos Números complexos
$\mathbb{H}$	Conjunto dos Quatérnios
$\mathbb{H}_0$	Conjunto dos Quatérnios Puros
$\mathcal{S}^3$	Conjunto dos Quatérnios Unitários
$\mathbb{D}$	Conjunto dos Números Duais
$\mathbb{H}\otimes\mathbb{D}$	Conjunto dos Quatérnios Duais
$\underline{\mathcal{S}}$	Conjunto dos Quatérnios Duais Unitários
$i,\ j,\ k$	Unidades Imaginárias
ε	Unidade Dual
$Re(a)$, $Im(a)$	Partes Real e Imaginária de a
$\mathcal{P}(a)$, $\mathcal{D}(a)$	Partes Primária e Dual de a
vec_n	Transformação isomórfica se algum espaço para o espaço vetorial de dimensãon
$\overset{+}{H}_4$,$\overset{-}{H}_4$	Operadores de Hamilton para produtos entre quatérnios na forma matricial
$\overset{+}{H}$, $\overset{-}{H}$	Operadores de Hamilton para produtos entre quatérnios duais na forma matricial
C_4	Operador para conjugar um quatérnio na forma matricial
C_8	Operador para conjugar um quatérnio dual na forma matricial
$T(a)$	Transformação afim de a
$\mathcal{F}$	Sistema de Coordenadas
p, q	Quatérnios
p_0, q_0	Quatérnios puros
$\underline{p}$, $\underline{q}$	Quatérnios Duais
ϕ	Ângulo de rotação de um quatérnio unitário
t	Translação de Quatérnios
r	Rotação de quatérnios
J	Jacobiana
θ	Vetor de Juntas
Θ	Vetor de Juntas aumentado
I_m	Matriz identidade de dimensão $m \times m$
$\underline{x}_{rel}$	Quatérnio da posição relativa
$\underline{x}_{abs}$	Quatérnio da posição absoluta
λ	Fator de amortecimento
K	Ganho do controlador

1 INTRODUÇÃO

"No sensible decision can be made any longer without taking into account not only the world as it is, but the world as it will be"

Isaac Asimov

A robótica é definida como a ciência de se perceber e manipular o mundo através de dispositivos mecânicos controlados por computadores. Em outras palavras, robôs são agentes que utilizam alguma forma de sensoriamento para receber informações acerca do mundo ao seu redor e a partir de tais informações conseguem realizar uma vasta gama de atividades, como planejar e mapear rotas e movimentos, manipular objetos ou até mesmo interagir com outros agentes (pessoas ou robôs) [7, 1].

Historicamente, pode-se retraçar o termo robô para a peça *Rossum's Universal Robots*, do autor checo Karel Capek, em que máquinas de aparência humanoide eram usadas como força de trabalho. Notavelmente, novelistas e artistas do último século se apegaram a ideia introduzida por Capek e robôs passaram ser uma das imagens mais características da ficção científica, afinal, quem não se lembra da adorável ROSE, no desenho infantil *Os Jetsons,* ou do R2D2 de Guerra nas Estrelas? De Asimov a Kubrick a robótica passou a fazer parte do imaginário de uma sociedade que muitas vezes ainda não percebeu que o futuro em que máquinas e humanos convivem e cooperam já é realidade.

Inicialmente, robôs foram criados como agentes a atuarem em ambientes industriais, muitas vezes executando tarefas repetitivas e de baixa complexidade. Entretanto, com o desenvolvimento da mecânica, eletrônica e de campos como cibernética e controle as fábricas perderam a exclusividade nesse campo e a robótica foi para hospitais, para a vida doméstica e até mesmo para o espaço![8]

Hoje se vê um notável desenvolvimento na robótica e em áreas correlatas — teoria de controle moderno, sensores de alta precisão, computadores com grandes capacidades de processamento são só algumas das tecnologias que permitiram que a robótica atingisse novos patamares. A Fig. 1.1, por exemplo, mostra algumas das diversas subdivisões que a robótica pode assumir.

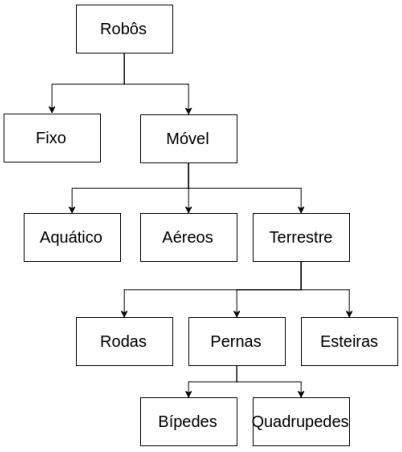


Fig. 1.1: Algumas subdivisões da robótica

Estes diferentes tipos de robôs podem realizar as mais variadas tarefas e, intimamente ligadas a estas tarefas, tem-se a atuação de controladores. Controladores podem ser modelados por diversas técnicas, assim como as variáveis a serem controladas. Na robótica é muito comum se controlar a pose e a trajetória de efetuadores utilizando modelos matemáticos e, dentre os meios de se realizar o modelamento, tem-se a descrição algébrica de estruturas conhecidas como quatérnios duais, em particular quatérnios duais unitários [2, 6]. Tais estruturas fazem parte da álgebra de Clifford, englobando os conceitos de quatérnios e de números duais, de modo que possuem oito dimensões e são capazes de descrever de maneira acoplada tanto um movimento de translação quanto de rotação no espaço tridimensional, evitando problemas com singularidades como o Bloqueio do Cardan[1].

Ainda pensando em como a robótica se insere na sociedade, pode-se notar a necessidade de robôs cooperarem com seres humanos ou entre si. Tarefas como manipular objetos podem ser facilitadas quando múltiplos agentes trabalham em conjunto. Assim, introduz-se a técnica de controle no espaço cooperativo, no qual o controle não é aplicado diretamente no efetuador de um robô, mas sim no espaço dividido entre os agentes em questão. Esse conceito pode ser expandido para espaço cooperativo dual quando aplicado a descrições com quatérnios duais [2, 6].

Finalmente, remetendo-se novamente a Fig. 1.1, nota-se que há uma subdivisão da robótica móvel que leva aos robôs humanoides. Estes são robôs bípedes de aparência antropomórfica, que, dado um mundo construído por humanos, tem grande facilidade de atuação e adaptação ainda que as tecnologias nessa área ainda estejam em desenvolvimento.

Assim, o presente trabalho se propõe juntar os diversos conceitos introduzidos nesta seção e criar uma estrutura em que robôs humanoides consigam atuar em conjunto na realização de uma tarefa. Para a modelagem do problema, a movimentação dos efetuadores é descrita a partir da matemática dos quatérnios duais e utiliza controladores no espaço cooperativo. Além disso, foi proposto um método de comunicação que não utilize uma rede sem fio, mas em seu lugar um método de comunicação visual.

[1] Bastante conhecida pelo seu nome em inglês, Gimbal Lock. Ocorre quando existem três eixos de rotação que se alinham fazendo com que se perca um grau de liberdade. Um exemplo famoso de problema com o Bloqueio de Cardan ocorreu na missão lunar Apollo 11 [9].

1.1 MOTIVAÇÃO

A robótica cooperativa se mostra necessária a partir do momento que um agente sozinho não consegue realizar uma tarefa. Por exemplo, a manipulação de objetos com um único braço pode ser complicada dadas as dimensões do mesmo — objetos muito grandes em que só há um ponto de apoio em sua extremidade podem se partir. Outra necessidade de cooperação entre agentes é na manipulação fina de um objeto, em que é muito mais fácil controlar sua direção, por exemplo, com os dois braços de um humanoide.

Motiva-se também a necessidade de controladores em espaço cooperativo quando robôs realizam tarefas em conjunto com seres humanos, nesse cenário é preciso controlar a posição do objeto, além de ser interessante criar métodos de comunicação em que não se precise de uma estrutura de rede. Seres humanos, assim como alguns dispositivos mecânicos, não conseguem se conectar a alguma rede.

1.1.1 RoboCup

Ainda no cenário de cooperação entre robôs, têm-se diversos eventos que visam promover avanços na área, dentre eles pode-se destacar a RoboCup.

O evento, que surgiu na década de 1990, tem como objetivo geral o desenvolvimento de um time de futebol formado por robôs e que seja capaz de competir e vencer um time de seres humanos em uma partida oficial da FIFA. Desse modo, a competição engloba diversas categorias que promovem diferentes desafios, como por exemplo, desenvolvimento de movimentos estáveis, construção de plataformas robustas, percepção de *features* do campo, localização e mapeamento simultâneos ou desenvolvimento e planejamento de comportamentos autônomos complexos.

Visando uma aproximação com o mundo real, o futebol se prova um experimento rico o suficiente para fomentar avanços nas pesquisas em robótica que buscam uma maior aplicabilidade prática. De fato, pode-se caracterizar a partida como um fragmento do mundo real, visto que jogadores podem ser definidos como agentes situados em um ambiente sobre o qual estão programados para atuar. Assim observa-se que dentro de um time tais agentes devem tanto cooperar quanto competir entre si, o ambiente é dinâmico e não completamente previsível, mesmo que todas as variáveis sejam conhecidas, é possível realizar uma avaliação objetiva da atuação do robô por meio do placar das partidas e fatores aleatórios podem ser desconsiderados, visto que os experimentos pode ser repetidos diversas vezes[10].

Dentro da RoboCup, pode-se enquadrar este trabalho na categoria SPL (Standard Platform League). Nesta categoria é utilizada uma plataforma pré-definida, o NAO, da Aldebaran Robotics, não havendo, portanto, preocupações com a fabricação mecânica do robô. Desse modo, é muito maior o foco no desenvolvimento de rotinas para o comportamento e coordenação entre os robôs, fazendo com que, cada vez mais, o jogo se assemelhe a uma partida de futebol jogada por humanos. Com esse intuito a cada ano são propostos novos desafios que tornam o cenário mais realista: partidas outdoor e detecção de apitos sendo alguns exemplos [11].

Em 2016 a SPL Robocup propôs o desafio “no wifi challenge”. Durante as partidas tradicionais da

RoboCup a coordenação de robôs de um mesmo time é feita através do envio de pacotes de dados via uma rede Wifi, entretanto, nem sempre há a possibilidade de se criar esta rede e, portanto, é proposto que se criem protocolos para a comunicação sem o uso de uma estrutura de redes, via gestos ou sons. Novamente, podemos enquadrar tal desafio no objetivo geral de tornar a partida mais parecida com o mundo real: a comunicação feita por seres humanos é feita através de gestos e sons, assim, deseja-se que os robôs interajam com outros agentes da mesma maneira.

1.1.1.1 Desafio “No Wifi”1.1.1.1

O desafio “No WiFi” proposto para a SPL 2016 tem como objetivo simular a comunicação entre robôs em uma partida de futebol, de forma que a transmissão de dados seja o mais próximo o possível daquela utilizada por humanos em uma partida real. Como foi estipulado nas regras dos Desafios Técnicos [12], tal comunicação poderia utilizar sons, visão computacional ou qualquer combinação entre eles.

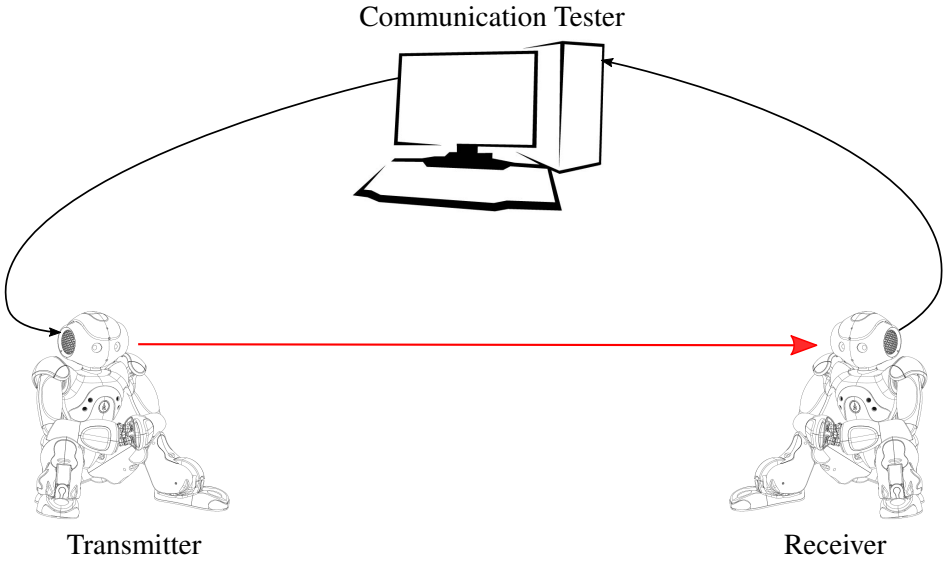


Fig. 1.2: Configuração física do Desafio. Setas em preto representam conexões via ethernet. Em vermelho tem-se a comunicação sem estrutura de redes WiFi entre os dois robôs. (*adaptado: http://doc.aldebaran.com/*)

O desafio é composto por dois robôs e um computador central como mostrado na Fig. 1.2. Um robô é designado como Transmissor e o outro como Receptor. Ambos os robôs são conectados em um computador por meio de conexão ethernet. O Receptor é então colocado na posição 1 demonstrada na Fig. 1.3, enquanto o Transmissor pode ser posicionado nos pontos 2 ou 3. As distâncias entre as posições 1 e 2 e entre as posições 1 e 3 são de aproximadamente 4 e 8 metros, respectivamente.

Após o correto posicionamento, os robôs devem se conectar com o computador central via TCP/IP. O transmissor deve então aguardar a conexão do software utilizado no desafio, denominado por CommsTester, o qual irá enviar uma mensagem contendo dois inteiros representando uma posição cartesiana (x, y) em que $-4500 \leq x \leq 4500$ e $-3000 \leq y \leq 3000$. Esta mensagem deve ser enviada para o receptor via comunicação sem estrutura de rede, que por fim deve retornar a mensagem

para o CommsTester e apontar para a posição aproximada no campo real. O tempo entre o envio da mensagem pelo CommsTester e o seu retorno não deve ser maior que 15 segundos.

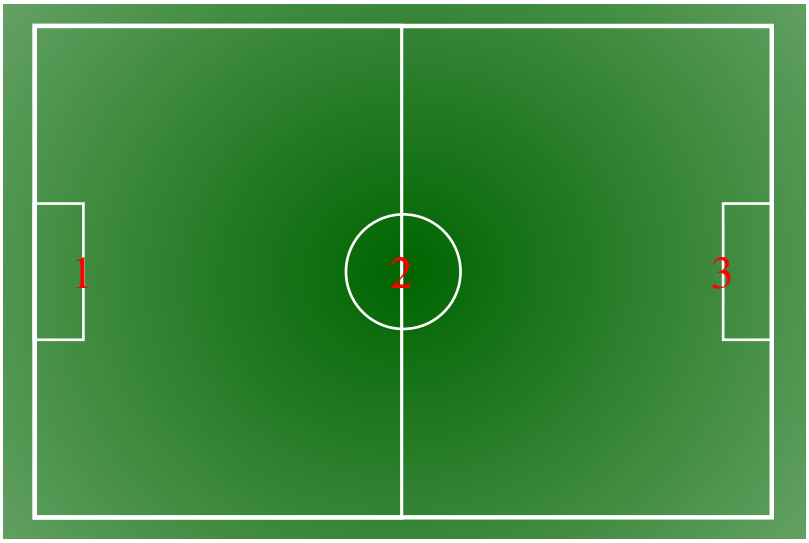


Fig. 1.3: Posicionamento dos robôs para a realização do desafio. Mensagens transmitidas com sucesso das posições 2 e 3 conferem 1 e 2 pontos, respectivamente.

1.2 OBJETIVOS

Visando um cenário em que robôs estão cada vez mais inseridos no mundo humano, é interessante imaginar maneiras em que eles possam atuar. Em um cenário doméstico, por exemplo, um robô humanoide pode atuar levando uma bandeja, um de seus braços poderá se coordenar com o outro de modo que se equilibre objetos, como um prato ou um copo em cima dessa bandeja. Esse é um cenário simples que exemplifica colaboração entre dois manipuladores.

Visando aprimorar ainda mais uma aplicação para robôs humanoides inseridos no mundo humano, pode-se pensar em um cenário em que um robô atue em conjunto com uma pessoa, ou mesmo com outro robô. Nesse cenário se faz necessária a comunicação entre os agentes, e principalmente visando a atuação com seres humanos, essa comunicação deverá ocorrer por meios visuais ou sonoros.

Desse modo se torna interessante a criação de um sistema que englobe os aspectos tanto de comunicação quanto de cooperação entre robôs, assim, visando auxiliar na construção de tal sistema, utilizando o cenário de uma aplicação doméstica e a proposta do desafio da RoboCup, esse projeto tem como objetivo criar um conjunto de ferramentas que possa ser utilizado para uma tarefa entre dois robôs.

1.3 CONTRIBUIÇÕES

Como contribuição esse trabalho primeiramente introduz uma interface para a captura de vídeos com o robô humanoide NAO além da disponibilização de um banco de dados com imagens do robô em diversas posições para utilização em treinamentos. Foi também introduzido um método de codificação de mensagem em que posições de um campo de futebol foram transformadas em números octais e transmitidos a partir de padrões em três LEDs diferentes.

Foi criado também um sistema utilizando diversas ferramentas de simulação em que a partir da matemática dos quatérnios duais foi possível controlar os braços de um NAO simulado, inclusive para a realização de tarefas cooperativas entre ambos manipuladores. Finalmente, foi criado um sistema para que tais tarefas pudessem ser implementadas em um robô físico.

Todos as ferramentas criadas se encontram disponíveis em m repositório do GitHub[2].

1.4 APRESENTAÇÃO DO MANUSCRITO

Este manuscrito introduz, no Capítulo 2, conceitos fundamentais para a realização do projeto, como técnicas de visão computacional, matemática dos quatérnios e temas relativos a robótica e controle. No Capítulo 3 é feita a descrição das principais ferramentas utilizadas, já nos Capítulos 4 e 5 é descrita a metodologia para a criação de ferramentas de comunicação visual e para o modelamento e validação dos controladores. Os Capítulos 6 e 7 apresentam os resultados obtidos nas seções de desenvolvimento e finalmente, o Capitulo 8 mostra as conclusões deste trabalho, além de apresentar maneiras de expandi-lo a aplicações para trabalhos futuros.

[2]https://github.com/lara-unb/TGs

2 FUNDAMENTAÇÃO

"Don't Panic"

The Hitchhiker's Guide to the Galaxy

2.1 INTRODUÇÃO

A robótica é uma área vasta e sistemas robóticos em geral são complexos, de modo que para atuar nesses sistemas faz-se necessária a sinergia de diversos campos da engenharia e da matemática.

Ao se analisar um robô, ou algum outro sistema de controle, irá se notar a presença de sensores (como câmeras, sonares, microfones, etc.), normalmente digitais, que transmitem os dados para um computador, um microcontrolador ou alguma outra variação de hardware. Embarcado neste hardware deverá haver algum software que processe os dados. Pensando no sistema robótico, os dados sensoreados poderão levar a alguma ação específica, por exemplo, dada uma leitura de um *encoder* de um motor pode requerer a atuação nesse motor ou, dado um objeto detectado por um sistema de visão computacional, poderá ser definido que o robô deva andar para o objeto. Além disso, por trás de todo o controle, todas as ferramentas computacionais e toda atuação mecânica existe uma teoria matemática.

Visando introduzir conceitos utilizados para realização do projeto, este capítulo faz uma breve apresentação dos fundamentos matemáticos e técnicos utilizados no desenvolvimento. Assim primeiramente será apresentada a plataforma utilizada: o humanoide NAO e em seguida serão discutidos conceitos sobre visão computacional, principalmente aqueles relacionados a detecção de objetos. Serão então apresentados conceitos em relação a matemática dos quatérnios duais e, ao final, será feita uma introdução quanto a robótica e controle.

2.2 VISÃO COMPUTACIONAL

O olho e o cérebro humano, trabalhando de forma conjunta, são capazes de executar uma vasta gama de tarefas envolvendo detecção de objetos e reconhecimento de padrões. Replicar estas tarefas de modo a serem implementadas por algoritmos é o desafio a ser resolvido pelas técnicas de Visão Computacional.

Visão Computacional pode ser descrita como a construção de descrições explícitas e claras de objetos em uma cena [13], o que a torna diferente de técnicas de processamento de imagem. Enquanto o processamento de imagens é um processo onde a entrada e a saída são imagens, a visão computacional emula a visão humana, recebendo como entrada imagens, porém tendo como saída uma interpretação total ou parcial desta cena. Aplicações de visão computacional podem, entretanto, conter algoritmos de processamento de imagens. Como especificado por Gonzales et al [14], os pro-

cessos que vão desde o processamento de imagens até aplicações de Visão Computacional podem ser divididos em três níveis:

- **Processos de baixo nível:** envolvem operações primitivas como mudanças estruturais, eliminação de ruídos ou melhorias de contraste, saturação, etc.
- **Processos de nível médio:** operações de segmentação como reconhecimento de objetos e classificação de dados.
- **Processos de alto nível:** estão relacionados com tarefas cognitivas diretamente relacionadas à visão humana.

Em se tratando de robótica, técnicas de visão computacional são ferramentas poderosas de sensoriamento, dado a grande quantidade de informações que podem ser obtidas em uma única imagem.

2.2.1 Detecção de Objetos

O reconhecimento de objetos é a aplicação do processamento de imagens e da Visão Computacional para que, havendo algum conhecimento prévio da aparência de um objeto, seja possível determinar sua presença em uma imagem, além de sua posição [15].

É notório que esta é uma área que vem ficando cada vez mais em evidência nas últimas décadas, principalmente dada a significativa melhora nas tecnologias digitais e crescente demanda de sistemas e aplicativos capazes de realizar algum tipo de reconhecimento. De fato, na literatura sobre o tema é possível encontrar diversos algoritmos, já bem estruturados, além de bibliotecas que possuem implementações já encapsuladas, prontas para serem aplicadas em algo mais específico. Um exemplo bastante famoso é a biblioteca OpenCV (Open Source Computer Vision Library) [16], utilizada no presente trabalho.

Dentro do OpenCV podemos encontrar implementações de diversas técnicas já bem conhecidas, como é o caso das funções para o Template Matching e Haar Cascade, que serão o foco desta seção.

2.2.1.1 Template Matching

Os termos *Template* e *Matching* carregam significados importantes no entendimento de tal técnica. *Template* pode ser definido como um padrão desenvolvido com o propósito de servir como modelo para algo a ser criado, isto é, um protótipo. *Matching* é o ato de comparar similaridade entre objetos.

A técnica de template matching consiste em encontrar pequenas partes de uma imagem que correspondem a um *template* ou a uma *feature* robusta. Existem diversas implementações deste processo, com diferentes velocidades de execução e acurácia. A técnica mais simples utilizada em visão computacional consiste em percorrer a imagem pixel a pixel e calcular a diferença quadrática entre o *template* e uma seção da imagem original. Após percorrer toda a imagem, deve-se então escolher as coordenadas que possuírem o menor resultado no cálculo de diferença.

$$R(x,y) = \sum_{x'y'} \left(T\left(x', y'\right) - I\left(x + x', y + y'\right) \right) \tag{2.1}$$

De acordo com [16] o cálculo da diferença quadrática para um dado ponto inicial da imagem original é mostrado em (2.1), em que (x, y) denota este ponto. Tem-se ainda que $T(x', y')$ é o valor do pixel de coordenada (x', y') do template e que $I(x, y)$ é o valor do pixel de coordenada (x, y) da imagem original.

2.2.1.2 Haar Cascade

O Haar Cascade é uma técnica inicialmente proposta por Paul Viola e Michael Jones [17], que buscavam resolver de modo rápido e robusto o problema da detecção de objetos em uma imagem. Para solucionar o problema, o método implementado tomou como base a classificação de descritores baseados em *features* do tipo haar como descrito por Papageorgiou et al [18]. Nesse método são calculadas as somas dos valores dos pixeis em janelas retangulares na imagem, e subsequentemente é feita a diferença entre regiões adjacentes, de modo a categorizar a imagem em subseções. Cada uma dessas subseções será considerada uma feature que pode ser classificada por sua posição em uma região de interesse, formato ou até mesmo pelo seu tamanho. Entretanto, estas são consideradas classificadores fracos, sendo detectados muito mais *features* do que pixeis em uma imagem. Desse modo o que é proposto é uma junção desses classificadores fracos em classificadores melhores, para isso é utilizada a técnica de aprendizado de máquina AdaBoost. A partir de uma análise das amostras negativas e positivas o classificador irá criando *features* mais complexas ou descartando resultados irrelevantes, como pode ser visto na figura 2.1.

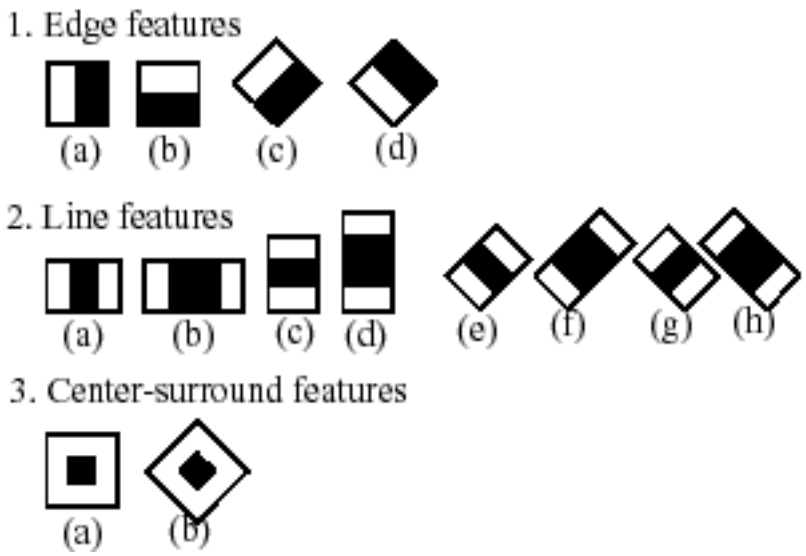


Fig. 2.1: Haar Features. *(fonte: opencv.org)*

Após as *features* mais complexas serem definidas, o classificador inicia um processo de atribuição de pesos, de modo que uma cascata com níveis cada vez mais complexos é formado. Esse processo é descrito como o treinamento do Haar Cascade.

2.3 CONCEITOS MATEMÁTICOS PARA QUATÉRNIOS DUAIS

O presente trabalho utiliza a descrição de quatérnios duais para implementar diversos controladores. Esta seção introduzirá, inicialmente, o conjunto dos números complexos, e em seguida a matemática por trás dos quatérnios, números duais e subsequentemente quatérnios duais.

2.3.1 Números Complexos

Durante a história, diversos matemáticos se depararam com o problema de se resolver equações quadráticas e, principalmente, cúbicas que não tinham soluções aparentes tendo em vista que se apresentavam radicais negativos. Foi no século XVI que Girolamo Cardano tentou solucionar o seguinte problema:

> "Encontrar dois números cuja a soma fosse igual a 10 e o seu produto fosse igual a 40."

O que eventualmente motivou a criação da unidade imaginária $i = \sqrt{-1}$ e a definição dos números complexos [19], dados por uma por uma parte real e uma parte imaginária como em

$$z = x + yi. \tag{2.2}$$

De fato, a partir da introdução dos números complexos na matemática e do subsequente entendimento físico e geométrico desses números muitas propriedades foram descobertas e estas foram aplicadas nos mais diversos campos da ciência e engenharia.

Formalmente é possível definir o conjunto dos números complexos sendo parte de $\mathbb{R}^2$, como mostrado por

$$\mathbb{C} \triangleq \left\{ x + yi | x, y \in \mathbb{R} \, e \, i^2 = -1 \right\}. \tag{2.3}$$

A seguir são definidas as operações de adição e multiplicação, de modo que se $z_1 = x_1 + y_1$ e $z_2 = x_2 + y_2$ tem-se a adição como mostrada em

$$z_1 + z_2 = (x_1 + x_2) + (y_1 + y_2)\, i \tag{2.4}$$

e a multiplicação em

$$z_1 z_2 = (x_1 x_2 - y_1 y_2) + (x_1 y_2 + y_1 x_2)\, i. \tag{2.5}$$

Além disso, ainda considerando $z = x + yi$ podem ser definidas a função dada por

$$Re(z) = x \tag{2.6}$$

denotando a parte real de um número complexo e a função dada por

$$Im(z) = y \tag{2.7}$$

denotando sua parte imaginária. Sendo definido também um número real como aquele em que a parte imaginária é zero e um número imaginário puro como aquele cuja parte real é zero.

Outros elementos e operações importantes no grupo dos números complexos são o conjugado, a equivalência, a inversa, a norma, a identidade e o elemento nulo, dados respectivamente por

$$\overline{z} = x - yi, \tag{2.8}$$

$$z_1 = z_2 \Longleftrightarrow x_1 = x_2 \, e \, y_1 = y_2, \tag{2.9}$$

$$z^{-1} = \frac{\overline{z}}{z\overline{z}}, \tag{2.10}$$

$$|z| = \sqrt{x^2 + y^2}, \tag{2.11}$$

$$z1 = 1z = z \tag{2.12}$$

$$z0 = 0z = 0 \tag{2.13}$$

Finalmente, pode-se comentar sobre a representação geométrica dos números complexos, que pode ser dada tanto pela forma polar no plano real, como por um vetor $\mathbb{R}^2$ no plano de Argand-Gauss (em que se tem o eixo ordenado como o eixo imaginário e o eixo das abcissas como o eixo real). Sendo a forma polar dada por

$$z = |z| \left(cos(\theta) + sen(\theta)\, i\right). \tag{2.14}$$

Considerando as propriedades do conjunto de números complexos e a representação no plano de Argand-Gauss, é intuitivo se pensar na possibilidade de serem realizadas transformações como rotações ou translações no plano. De fato, as propriedades geométricas desses números foi um dos fatores motivantes para a evolução de diversas áreas da engenharia como no estudo de circuitos, processamento de sinais, aviação ou, como é o caso do presente trabalho, robótica.

2.3.2 Quatérnios

Visando criar uma generalização dos números complexos em que se pudesse representar transformações no espaço tridimensional, o matemático William Hamilton tentou por muito tempo desenvolver um sistema hipercomplexo dado por uma tripla de números complexos. Esse esforço, apesar

de não ter gerado os resultados que Hamilton desejava, o levaram na direção que resultou no sistema quadridimensional dos quatérnios [20] [21].

Denota-se o sistema algébrico criado por Hamilton como $\mathbb{H}$, em que quatérnios podem ser definidos como em

$$\mathbb{H} \triangleq \left\{ a + bi + cj + dk \,|\, a, b, c, d \in \mathbb{R}\,,\, i^2 = j^2 = k^2 = ijk = -1 \right\}. \tag{2.15}$$

Dada a definição de quatérnios, é importante notar que estes números perdem a propriedade de comutação, ou seja, $ij \neq ji$. A multiplicação dos quatérnios pode ser vista na tabela de multiplicação como na Tab. 2.1.

Tab. 2.1: Tabela de Multiplicação para Quatérnios

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Como visto, os quatérnios são uma expansão dos números complexos, de modo que eles possuem uma parte real e uma parte imaginária. Tomando o quatérnio $q = a+bi+cj+dk$, tem-se as definições

$$Re(q) = a \tag{2.16}$$

e

$$Im(q) = bi + cj + dk \tag{2.17}$$

Para simplificar a notação, considera-se que o quatérnio é isomorfo a um vetor de números reais quanto a adição, assim, considerando $v = bi + cj + dk$, a parte imaginária será dada pelo vetor $vec_3 v = [b\ c\ d]^T$ de modo que o quatérnio será dado por $q = a + v$. Dessa forma, tomando os quatérnios $q_1 = a_1 + v_1$ e $q_2 = a_2 + v_2$ podem ser definidas algumas de suas propriedades [22]. Define-se a adição dos quatérnios em

$$q_1 + q_2 = (a_1 + v_1) + (a_2 + v_2) = (a_1 + a_2) + (v_1 + v_2). \tag{2.18}$$

É definido o produto entre os quatérnios como

$$\begin{aligned}
q_1 q_2 =& (a_1 + v_1)(a_2 + v_2) \\
=& (a_1 a_2 - v_1.v_2) + (a_1 v_2 + a_2 v_1 + v_1 \times v_2) \\
=& a_1 a_2 - (b_1 b_2 + c_1 c_2 + d_1 d_2) + (a_1 b_2 i + a_1 c_2 j + a_1 d_2 k + a_2 b_1 i + a_2 c_1 j + a_2 d_1 k) + \\
& (c_1 d_2 - d_1 c_2) i + (d_1 b_2 - b_1 d_2) j + (b_1 c_2 - c_1 b_2) k \\
=& (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) + (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2) i \\
& + (a_1 c_2 + c_1 a_2 + d_1 b_2 - b_1 d_2) j + (a_1 d_2 + d_1 a_2 + b_1 c_2 - c_1 b_2) k.
\end{aligned} \tag{2.19}$$

em que $v_1.v_2$ será o produto interno e $v_1 \times v_2$ o produto vetorial entre v_1 e v_2.

E se define a igualdade entre quatérnios como em

$$q_1 = q_2 \iff a_1 = a_2,\ b_1 = b_2,\ c_1 = c_2\ e\ d_1 = d_2. \tag{2.20}$$

Também são definidas, na álgebra dos quatérnios, as propriedades de conjugado, norma e inversa, como em

$$q* = a - v = a - bi - cj - dk, \tag{2.21}$$

$$\|q\|^2 = qq* = q * q = a^2 + b^2 + c^2 + d^2 \tag{2.22}$$

e

$$q^{-1} = \frac{q*}{\|q\|^2}. \tag{2.23}$$

Ainda utilizando a notação $q = a + v$, é possível definir quatérnios puros (denominado por $\mathbb{H}_0$), em que $a = 0$, sendo $vec_3 q = [b\ c\ d]^T$. Nota-se que nesse caso existe um mapeamento de um subespaço dos quatérnios $\mathbb{H}_0$ para o vetor real de três dimensões $\mathbb{R}^3$. Tal mapeamento pode ser dado por

$$vec_3 : \mathbb{H}_0 \to \mathbb{R}^3. \tag{2.24}$$

Essa propriedade em que se realiza o mapeamento entre dois grupos e se preserva as operações definidas em ambos é chamada de isomorfismo. Análogo ao mapeamento da (2.24), pode-se definir o mapeamento do grupo dos quatérnios para um vetor em $\mathbb{R}^4$, ou seja, o quatérnio q será dado pelo vetor $vec_4 q = [a\ b\ c\ d]^T$, definindo-se portanto a operação

$$vec_4 : \mathbb{H} \to \mathbb{R}^4. \tag{2.25}$$

Finalmente, utilizando-se o conceito de isomorfismo e o mapeamento definido em (2.25), pode-se definir uma estrutura relacionada ao produto de quatérnios que permita a comutação entre os termos, ou seja, tomando $q_1 = a_1 + b_1 i + c_1 j + d_1 k$ e $q_2 = a_2 + b_2 i + c_2 j + d_2 k$ tem-se que

$$vec_4\,(q_1 q_2) = \overset{+}{H}_4(q_1) vec_4 q_2 = \overset{-}{H}_4(q_2) vec_4(q_1), \tag{2.26}$$

em que os termos $\overset{+}{H}_4$ e $\overset{-}{H}_4$ são denominados operadores de Hamilton e podem ser definidos respectivamente como

$$\overset{+}{H}_4(q) = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} \tag{2.27}$$

e

$$\overset{-}{H}_4(q) = \begin{bmatrix} a & -b & -c & -d \\ b & a & d & -c \\ c & -d & a & b \\ d & c & -b & a \end{bmatrix}. \tag{2.28}$$

Estes operadores definem a forma vetorial da multiplicação entre quatérnios.

2.3.2.1 Quatérnios Unitários

Quatérnios unitários são um subgrupo dos quatérnios de extrema importância quando se deseja realizar rotação no espaço tridimensional. Este subgrupo é definido por

$$\mathcal{S}^3 \triangleq \{q \in \mathbb{H} \mid \|q\| = 1\}. \tag{2.29}$$

Os quatérnios unitários são definidos de modo que a parte imaginária representa um eixo de rotação, por exemplo $v = v_x\grave{\imath} + v_y j + v_z k$, e a parte real representa um ângulo de rotação em torno deste eixo, por exemplo ϕ.

Desse modo tem-se algumas propriedades, como a inversa ser igual ao conjugado e a representação do quatérnio na forma polar, dados respectivamente por

$$q^{-1} = \frac{q*}{\|q\|^2} \Rightarrow \|q\| = 1 \Rightarrow q^{-1} = q* \tag{2.30}$$

e

$$r = cos\left(\frac{\phi}{2}\right) + sen\left(\frac{\phi}{2}\right)v. \tag{2.31}$$

Nota-se que r representa a rotação do ângulo ϕ em torno do eixo v.

2.3.3 Números Duais

Desde a criação dos números complexos por Cardano no século XVI, diversos matemáticos desenvolveram novas definições relacionadas a estes números. No século XIX o matemático William Clifford desenvolveu os números duplos e o matemático Eduard Study os números duais, de modo que em conjunto com os números complexos ordinários, pode-se criar uma definição generalizada dos números complexos, seguindo a forma

$$z = x + yE \tag{2.32}$$

em que E representa a unidade imaginária ($i^2 = -1$), dupla ($e^2 = 1$) ou dual ($\varepsilon^2 = 0$) [23].

Nesta seção, será mantido o foco nos números duais, definidos como

$$\mathbb{D} \triangleq \left\{x + y\varepsilon \mid a, b \in \mathbb{R},\ \varepsilon^2 = 0\right\}. \tag{2.33}$$

Nota-se que, apesar de $\varepsilon^2 = 0$, tem-se $\varepsilon \neq 0$. Sendo assim, ε é uma entidade algébrica abstrata não devendo ser considerada um número real.

Dados os números duais $\underline{z_1} = x_1 + y_1\varepsilon$ e $\underline{z_2} = x_2 + y_2\varepsilon$ as propriedades de adição, multiplicação e igualdade são definidas respectivamente como

$$\underline{z_1} + \underline{z_2} = (x_1 + x_2) + (y_1 + y_2)\varepsilon, \tag{2.34}$$

$$\underline{z_1 z_2} = x_1 x_2 + (x_1 y_2 + x_2 y_1)\varepsilon \tag{2.35}$$

e

$$\underline{z_1} = \underline{z_2} \iff x_1 = x_2 \, e \, y_1 = y_2. \tag{2.36}$$

Ainda de maneira análoga aos números complexos ordinários, pode-se separar os números duais em suas partes primária

$$\mathcal{P}(z) = x \tag{2.37}$$

e dual

$$\mathcal{D}(z) = y\varepsilon. \tag{2.38}$$

Finalmente, é interessante também a definição de

$$\underline{z}^{-1} = \frac{1}{x^2}(x - y\varepsilon), \; x \neq 0, \tag{2.39}$$

em que se tem a operação inversa, na qual é importante notar que a parte primária deve ser necessariamente diferente de zero de modo que a inversa possa existir.

2.3.4 Quatérnios Duais

Como definido nas seções anteriores, os quatérnios, especificamente os quatérnios unitários, descrevem de maneira efetiva rotações no espaço. Entretanto, deixam a desejar na descrição completa de um movimento, visto que com a estrutura criada por Hamilton não é possível descrever uma translação. Com a motivação de obter uma generalização que representasse o movimento de forma mais abrangente, em 1882, Clifford apresentou seu trabalho sobre biquatérnios (hoje chamados de quatérnios duais) [24].

Os quatérnios duais são definidos no espaço $\mathbb{R}^8$, como em

$$\mathbb{H}\otimes\mathbb{D} \triangleq \{(a + bi + cj + dk) + (x + yi + zj + wk)\varepsilon \mid a, b, c, d, x, y, z, w \in \mathbb{R}\}. \tag{2.40}$$

De modo que se juntam os conceitos de quatérnios com o conceito de números duais, criando então, uma estrutura capaz de representar transformações de translação e rotação de forma desacoplada [25, 26].

Nota-se que esta entidade algébrica é composta por uma quatérnio na parte primária e um na parte dual, de modo que dados dois quatérnios, p e s por exemplo, pode-se denotar o quatérnio dual como

$$\underline{q} = p + s\varepsilon. \tag{2.41}$$

Assim como nos grupos dos números complexos, duais e quatérnios, podem ser definidas as operações de adição, multiplicação e equivalência. Dados os quatérnios duais $\underline{q_1} = p_1 + s_1\varepsilon$ e $\underline{q_2} = p_2 + s_2\varepsilon$, as operações são definidas respectivamente em

$$\underline{q_1} + \underline{q_2} = (p_1 + p_2) + (s_1 + s_2)\varepsilon, \tag{2.42}$$

$$\underline{q_1 q_2} = p_1 p_2 + (p_1 s_2 + p_2 s_1) \tag{2.43}$$

e

$$\underline{q_1} = \underline{q_2} \iff p = p_2,\ s_1 = s_2. \tag{2.44}$$

Nota-se, particularmente, em (2.43) que a multiplicação de quatérnios duais retém a propriedade dos quatérnios de anticomutatividade. Apesar de termos com a unidade dual comutarem entre si, de modo que $\varepsilon i = i\varepsilon$, $\varepsilon j = j\varepsilon$ e $\varepsilon k = k\varepsilon$, multiplicações como $ij\varepsilon \neq ji\varepsilon$ não mantêm tal propriedade, logo, $\underline{q_1 q_2} \neq \underline{q_2 q_1}$.

Como dito, quatérnios duais são dados por uma combinação entre quatérnios e números duais, desse modo, podem ser definidas tanto as funções de parte primária e dual, quanto de parte real e imaginária. Assim, dado um quatérnio dual $\underline{q} = p + s\varepsilon$ e as definições de (2.16), (2.17), (2.37) e (2.38), é possível estendê-las para o conjunto dos quatérnios duais, como em

$$\mathcal{P}(\underline{q}) = p \ \ e \ \ \mathcal{D}(z) = s\varepsilon \tag{2.45}$$

e em

$$Re(\underline{q}) = Re(p) + \varepsilon Re(p) \ \ e \ \ Im(\underline{q}) = Im(s) + \varepsilon Im(s). \tag{2.46}$$

Ainda utilizando o quatérnio dual $\underline{q} = p + s\varepsilon$ pode ser definida a operação de conjugado como em

$$\underline{q}* = p^* + s^*\varepsilon. \tag{2.47}$$

De maneira análoga aos quatérnios, e utilizando-se a definição (2.43), tem-se que a norma pode ser obtida como

$$\begin{aligned}
\left\|\underline{q}\right\|^2 = & \\
\underline{qq}* &= pp* + (ps^* + sp^*)\varepsilon \\
&= \|p\|^2 + (ps^* + (s^*p)*)\varepsilon \\
&= \|p\|^2 + (\ (ax + by + cz + dz) + (-ay - bx - cw + dz)i + \\
&\quad (-az - cx - dx + bw)j + (-aw - dx - bz + cy)k \ + \ (ax + by + cz + dz) \\
&\quad + (ay + bx + cw - dz)i + (az + cx + dx - bw)j + (aw + dx + bz - cy)k\ )\varepsilon \\
&= \|p\|^2 + 2(ax + by + cz + dw)\varepsilon \\
&= \|p\|^2 + 2(vec_4 p \ \cdot \ vec_4 d)\varepsilon.
\end{aligned} \tag{2.48}$$

Nota-se que, dada a representação dos quatérnios por $vec_4 p = [a\ b\ c\ d]^T$ e $vec_4 s = [x\ y\ z\ w]^T$ pode-se utilizar o produto escalar para simplificar a definição de seminorma [27].

Finalmente, pode-se definir a operação inversa a partir da seminorma e conjugado, como em

$$\underline{q}^{-1} = \frac{\underline{q}^*}{\|\underline{q}\|^2},\ \|p\| \neq 0 \tag{2.49}$$

em que é importante se notar que só há definição se $\|p\| \neq 0$.

Pode-se analisar os quatérnios duais do mesmo modo que os simples, de modo que há uma relação de isomorfismo entre o quatérnio e um vetor, nesse caso em $\mathbb{R}^8$. Tal mapeamento pode ser dado por

$$vec_8 : \mathbb{H}\otimes\mathbb{D} \rightarrow \mathbb{R}^8. \tag{2.50}$$

Neste caso, tem-se que $vec_8 \underline{q} = [vec_4 p\ vec_4 s]^T = [a\ b\ c\ d\ x\ y\ z\ w]^T$. Ainda em analogia aos quatérnios simples, é possível se definir um quatérnio dual puro, ou seja, um quatérnio cuja parte imaginária seja zero. Assim se faz um mapeamento em $\mathbb{R}^6$ de modo que a operação vec_6 é definida como em

$$vec_6 : \mathbb{H}\otimes\mathbb{D} \rightarrow \mathbb{R}^6 \tag{2.51}$$

de modo que $vec_6 \underline{q} = [b\ c\ d\ y\ z\ w]^T$.

Voltando a forma vetorial $vec_8 \underline{q} = [a\ b\ c\ d\ x\ y\ z\ w]^T$, pode-se definir os operadores de Hamilton, de modo análogo ao que foi definido em (2.27) e (2.28), para os quatérnios duais. Aqui tais operadores serão chamados de $\overset{+}{H}$ e $\overset{-}{H}$ e são usados de modo a permitir a comutação entre os quatérnios duais, como em

$$vec_8\left(\underline{q_1 q_2}\right) = \overset{+}{H}\left(\underline{q_1}\right) vec_8\left(\underline{q_2}\right) = \overset{-}{H}\left(\underline{q_2}\right) vec_8\left(\underline{q_1}\right). \tag{2.52}$$

Utilizando mais uma vez o quatérnio dual $\underline{q} = p + s\varepsilon$, tem-se os operadores de Hamilton, $\overset{+}{H}$ e $\overset{-}{H}$, que podem ser definidos respectivamente como em

$$\overset{+}{H}(\underline{q}) = \begin{bmatrix} \overset{+}{H}_4(p) & 0_4 \\ \overset{+}{H}_4(s) & \overset{+}{H}_4(p) \end{bmatrix} \tag{2.53}$$

e

$$\overset{-}{H}(\underline{q}) = \begin{bmatrix} \overset{-}{H}_4(p) & 0_4 \\ \overset{-}{H}_4(s) & \overset{-}{H}_4(p) \end{bmatrix}. \tag{2.54}$$

Finalmente, é possível definir um operador adicional, denominado C_8, cuja finalidade é auxiliar na operação de conjugado quando se está utilizando a representação vetorial dos quatérnios duais. Desse modo, tem-se a definição de

$$vec_8\underline{q}* = C_8 vec_8\underline{q}. \tag{2.55}$$

Assim C_8 é definido como em

$$C_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{2.56}$$

2.3.4.1 Quatérnio Dual Unitário

Do mesmo modo que os quatérnios unitários, pode-se definir um subgrupo dos quatérnios duais: os quatérnios duais unitários. Os quatérnios duais unitários são particularmente uteis visto que descrevem completamente a movimentação de corpos rígidos no espaço utilizando somente oito parâmetros. Dado o quatérnio dual $\underline{q} = p + s\varepsilon$ (em que $p = a + bi + cj + dk$ e $s = x + yi + zj + wk$), pode-se colocar a condição de que $\left\|\underline{q}\right\| = 1$, de modo a restringi-lo a esfera unitária. Assim utilizando (2.48), pode-se obter as restrições de

$$\|p\| = \sqrt{a^2 + b^2 + c^2 + d^2} = 1 \tag{2.57}$$

e

$$vec_4 p \cdot vec_4 s = ax + by + cz + dw = 0. \tag{2.58}$$

Portanto, pode-se definir formalmente o conjunto dos quatérnios duais unitários como em

$$\underline{\mathcal{S}} \triangleq \left\{ \underline{q} \in \mathbb{H} \otimes \mathbb{D} \mid \|p\| = 1 \ \ e \ \ vec_4 p \cdot vec_4 s = 0 \right\}, \tag{2.59}$$

em que é interessante notar que a definição da inversa permanece como nos quatérnios unitários, como visto em

$$\underline{q}^{-1} = \frac{\underline{q}*}{\left\|\underline{q}\right\|^2} \Rightarrow \left\|\underline{q}\right\| = 1 \Rightarrow \underline{q}^{-1} = \underline{q}\,*\,. \tag{2.60}$$

Com este conjunto define-se também o grupo $Spin(3) \ltimes \mathbb{R}^3$, em que $\ltimes$ representa o produto indireto[1].

[1] Conceito em que um grupo é formado a partir de dois outros

2.4 CINEMÁTICA DE CORPOS RÍGIDOS

Um corpo rígido é definido como um conjunto finito de partículas cujas distâncias relativas entre qualquer um de seus pontos é invariante no tempo [28]. Visto que a deformação destes corpos é desprezada, a descrição geométrica de sua movimentação pode ser facilmente obtida através da composição de translações e rotações. Matematicamente, dadas as posições de duas partículas $a(t)$ e $b(t)$ durante um período de tempo, a distância relativa entre elas em um dado momento será como em

$$\|a(t) - b(t)\| = \|a(0) - b(0)\| = constante. \tag{2.61}$$

Os conceitos de movimento de um corpo rígido foram estudados, em 1800, por Michel Chasles e Louis Poinsot que introduziram, a partir do Teorema de Chasles, a ideia de que qualquer movimento pode ser descrito por uma rotação em torno de um eixo seguido por uma translação nesse mesmo eixo. Tal teoria foi mais tarde formalizada por Robert Ball como a Teoria Helicoidal[2] em que se utiliza conceitos da álgebra linear além dos grupos de Lie para descrever a pose de um corpo rígido [29].

De modo a exemplificar a descrição da pose de um corpo rígido no espaço tridimensional, toma-se a base ortonormal da Fig. 2.2, dada pelos eixos $\overrightarrow{x}, \overrightarrow{y}$ e $\overrightarrow{z}$. O eixo de coordenadas do corpo rígido será aplicado em seu centro de massa, no caso *O'*, e será dado pelo eixo de coordenadas $\overrightarrow{x}', \overrightarrow{y}'$ e $\overrightarrow{z}'$.

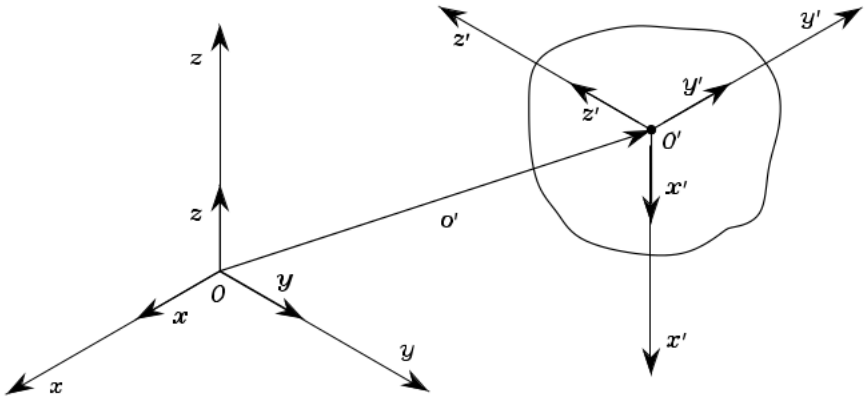


Fig. 2.2: Posição e rotação de um corpo rígido (*O*') em relação a uma base (*O*) [1]

Nota-se que o sistema do objeto *O'* é dado por um mapeamento da base *O,* que será do tipo $f : \mathbb{R}^n \to \mathbb{R}^m$. Este mapeamento pode ser tanto transformações de translação quanto rotação ou uma composição de ambas, não havendo, portanto, a possibilidade de reflexões ou deformações. Para isso, devem ser mantidas as propriedades da Eq. (2.61), já definida, além do produto vetorial,

$$f(v \times w) = f(v) \times f(w),\ v, w \epsilon \mathbb{R}, \tag{2.62}$$

que deverá ser preservado de modo a evitar reflexões [28].

[2] Em inglês, *Screw Theory*

Alguns dos métodos para descrever a pose de um objeto, são os ângulos de Euler e Quatérnios Unitários para rotações e Matrizes Homogêneas e Quatérnios Duais Unitários, para transformações de rotação e translação.

2.4.1 Transformações Afins

Pode-se definir uma transformação afim como um mapeamento entre dois espaços vetoriais de modo que sejam preservadas as operações de adição vetorial e multiplicação por escalar. Ou seja, tais transformações preservam proporções dos corpos que as sofrem. A equação

$$T(a) = L(a) + p_0 \tag{2.63}$$

mostra a estrutura de uma transformação afim, em que a transformação T aplicada a a será dada por uma transformação linear L somado a um ponto [30].

As transformações afins podem ser divididas, principalmente, em cinco grupos, como sumarizado a seguir [31].

- As **Translações** são definidas como a movimentação de todos os pontos de um objeto em um eixo. Essa transformação é definida como um mapeamento do tipo $\tau : \mathbb{R}^3 \to \mathbb{R}^3$ de modo que são preservadas todas as posições relativas das partículas, seu sentido e orientação.
- **Rotações** são transformações em que um objeto rotaciona em torno de um ponto, podendo ser definida como uma função φ que mapeia $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$. Desse modo são preservadas as distâncias entre os pontos, os ângulos em relação a origem, além do seu sentido.
- A **Mudança de Escala** é a transformação de altera as dimensões de um objeto, ou seja, a transformação é um mapa de $\mathbb{R}^3 \to \mathbb{R}^3$, de modo que $T(a) = Ka$, em que $K = diag(k_x, k_y, k_z)$.
- A transformação de **Cisalhamento** provoca uma deformação no objeto de modo que pelo menos uma direção se mantenha paralela, em outras palavras, esta é uma deformação em que pontos são multiplicados por constantes ao longo de um dos eixos do objeto.
- A **Reflexão** é a operação de rotação em que, dado um espaço de n dimensões, irá girar em torno de um eixo na dimensão $n + 1$ de modo a refletir o sistema coordenado do objeto. Para sistemas tridimensionais, tem-se que uma rotação em um sistema dextrogiro leva há um sistema levogiro[3].

A Fig. 2.3 mostra algumas das possíveis transformações afins a serem realizadas em um objeto.

[3]Dextrogiro é um sistema de referência que cumpre o sentido da regra da mão direita. O sentido levogiro está em contraposição a regra da mão direita.

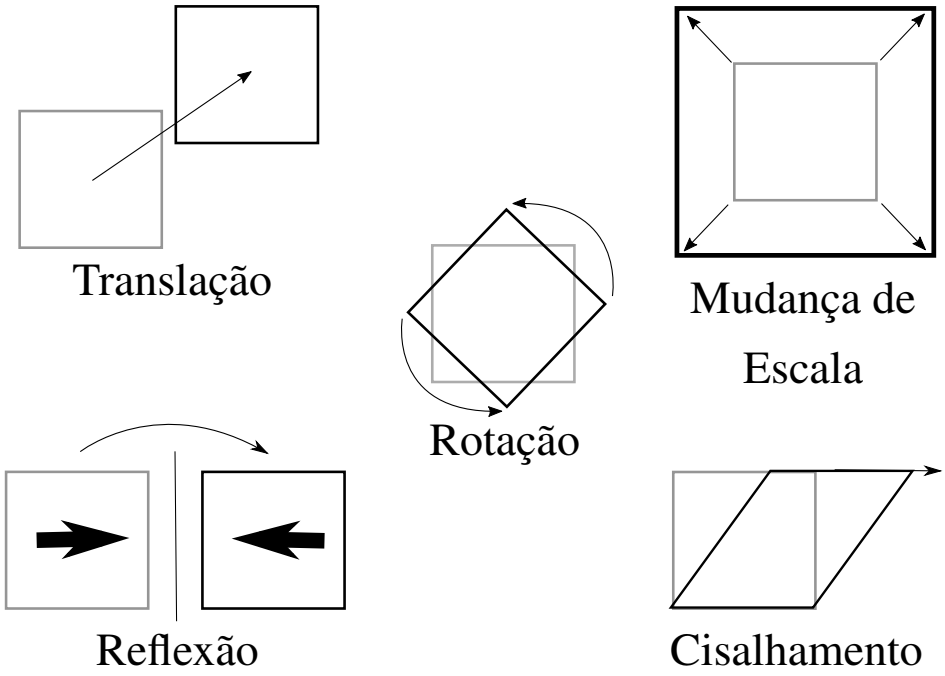


Fig. 2.3: Transformações Afins

Nota-se, entretanto, que, para corpos rígidos, há transformações afins que não podem ser fisicamente realizadas, principalmente visto que as propriedades de (2.61) e (2.62) devem ser mantidas. Assim, para este trabalho serão consideradas somente transformações de rotação e translação.

2.4.2 Matrizes Homogêneas

Uma matriz homogênea é uma matriz de dimensões $((n+1) \times (n+1))$ onde n é o número de dimensões do espaço no qual ela se aplica. De forma geral, uma matriz de transformação afim é dada como descrito na equação

$$T = \begin{bmatrix} X & Y \\ [0 \dots 0] & 1 \end{bmatrix} \tag{2.64}$$

em que X é uma matriz $(n \times n)$ e Y é um vetor $(n \times 1)$.

Para aplicar uma transformação em um espaço de n dimensões, basta multiplicar a matriz por um vetor coluna a, como mostrado em

$$a' = \begin{bmatrix} a'_1 \\ \vdots \\ a'_n \\ 1 \end{bmatrix} = \begin{bmatrix} X & Y \\ [0 \dots 0] & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ 1 \end{bmatrix}. \tag{2.65}$$

De forma análoga, tem-se a equação

$$a' = \begin{bmatrix} a'_x \\ a'_y \\ a'_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_{xx} & X_{xy} & X_{xz} & Y_x \\ X_{yx} & X_{yy} & X_{yz} & Y_y \\ X_{zx} & X_{zy} & X_{zz} & Y_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \\ 1 \end{bmatrix} \tag{2.66}$$

exemplificando uma transformação em um espaço tridimensional.

Pode-se definir também a multiplicação entre transformações afins, como mostrado em

$$T = T_1 T_2 = \begin{bmatrix} X_1 & Y_1 \\ [0 \dots 0] & 1 \end{bmatrix} \begin{bmatrix} X_2 & Y_2 \\ [0 \dots 0] & 1 \end{bmatrix} = \begin{bmatrix} X_1 X_2 & X_1 Y_2 + Y_1 \\ [0 \dots 0] & 1 \end{bmatrix}. \tag{2.67}$$

Nota-se que o resultado ainda é uma transformação afim.

Por fim, uma matriz afim é inversível, se e somente se, X for também inversível, e a sua inversa é dada como mostrado em

$$T^{-1} = \begin{bmatrix} X^{-1} & -X^{-1}Y \\ [0 \dots 0] & 1 \end{bmatrix}$$

Considerando as aplicações propostas neste trabalho, as únicas transformações afins a serem utilizadas são a translação e a rotação.

A translação utilizando matrizes homogêneas pode ser definida como mostrado em

$$A = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.68}$$

no qual d_x, d_y e d_z são distâncias a serem percorridas em x, y e z, respectivamente. Pode-se notar que a translação pura é uma transformação afim em que $X = I_{3x3}$, logo tal operação pode ser aplicada como especificado em (2.66).

A rotação em um espaço tridimensional pode ser descrita como uma matriz ortogonal[4] de determinante igual a 1. Pode-se definir três matrizes homogêneas de rotação distintas, como mostrado em

[4]Matriz quadrada onde a sua transposta é igual a sua inversa.

$$
\begin{aligned}
R_x &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sen\alpha & 0 \\ 0 & sen\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
R_y &= \begin{bmatrix} cos\beta & 0 & sen\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sen\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
R_z &= \begin{bmatrix} cos\gamma & -sen\gamma & 0 & 0 \\ sen\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}
\tag{2.69}
$$

no qual α, β e γ representam ângulos de rotação em torno dos eixos x, y e z, respectivamente.

Assim, pode-se representar, por exemplo, uma rotação em torno de x, seguida por uma rotação em torno de y, seguida por uma rotação em torno de z como mostrado em

$$
R = R_z R_y R_x = \begin{bmatrix} c\beta.c\gamma & -c\alpha.s\gamma + s\alpha.s\beta.c\gamma & s\alpha.s\gamma + c\alpha.s\beta.c\gamma & 0 \\ c\beta.s\gamma & c\alpha.c\gamma + s\alpha.s\beta.s\gamma & s\alpha.c\gamma + c\alpha.s\beta.s\gamma & 0 \\ -s\beta & s\alpha.c\beta & c\alpha.c\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.70}
$$

Note que, afim de simplificar a notação, o cosseno é representado por c e o seno por s.

É importante notar que a operação de rotação não é comutativa, ou seja, $R_z R_y R_x \neq R_x R_y R_z$. Portanto, a ordem de aplicação de rotações em sequência deve ser levada em conta ao se executar diferentes transformações.

2.4.3 Quatérnios Unitários

Como definido previamente, na Sec. 2.3.2.1, quatérnios unitários se mostram particularmente interessantes quando utilizados para descrever rotações no espaço tridimensional. Estes quatérnios de norma unitária são definidos pela (2.29), e dada sua propriedade de fechamento, para qualquer multiplicação sempre estarão definidos dentro do grupo $Spin(3)$.

Assim como nos ângulos de Euler e nas Matrizes Homogêneas, é feito o mapeamento das coordenadas espaciais para o quatérnio. No caso de quatérnios unitários tem-se a parte real indicando um ângulo (ϕ) que irá indicar uma rotação em torno de um eixo dado pela parte imaginária.

Com o intuito de ilustrar o problema de rotação de um ponto, considera-se o quatérnio puro $p_0^1 = p_x i + p_y j + p_z k$ (definido na Sec. 2.3.2). Quatérnios puros são geralmente utilizados para descrever uma translação ou um ponto no espaço, assim, o ponto mapeado por p_0^1 se encontra nas coordenadas $[p_x \ p_y \ p_z]^T \epsilon \mathbb{R}^3$ em relação a um sistema de coordenada $\mathcal{F}_1$. Além disso, define-se

o quatérnio unitário q. Deseja, então, aplicar uma rotação denotada por q no ponto p_0^1 de modo a levá-lo ao ponto p_1^1 também relativo ao sistema $\mathcal{F}_1$. Pode-se ver tal operação em

$$p_1^1 = q p_0^1 q^*. \tag{2.71}$$

De fato, o resultado desta operação é um quatérnio puro (ao se expandir a multiplicação a parte real irá se anular) e, além disso, a norma do vetor p_1^1 será dada pela equação

$$\left\|p_1^1\right\| = \|q\| \left\|p_0^1\right\| \|q^*\| = 1 \left\|p_0^1\right\| 1 = \left\|p_0^1\right\|, \tag{2.72}$$

em que se pode notar que será mantido o comprimento do vetor p.

A operação da (2.71) define uma rotação q do ponto p_0^1 no sistema de coordenadas $\mathcal{F}_1$ [32]. Caso se desejasse rotacionar $\mathcal{F}_1$ em torno do ponto p_0^1 de modo a levá-lo para um sistema de coordenadas $\mathcal{F}_2$, seria utilizada

$$p_0^2 = q^* p_0^1 q. \tag{2.73}$$

A Fig. 2.4 ilustra tanto o movimento descrito pela (2.71) quanto pela (2.72).

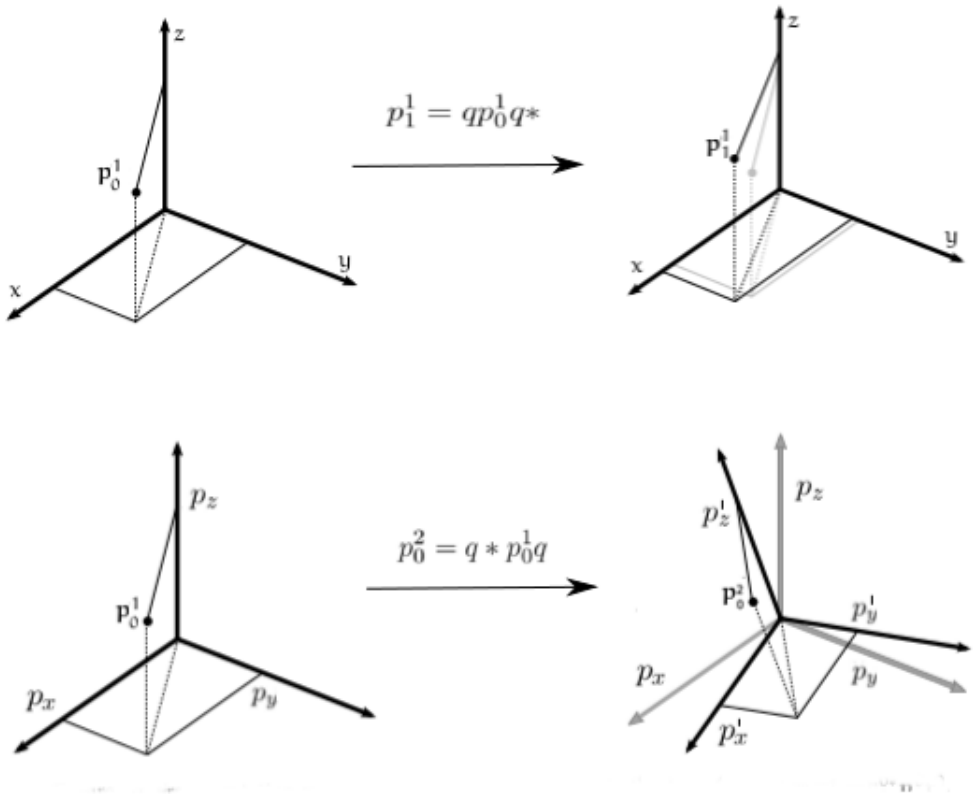


Fig. 2.4: Rotações de um ponto em relação ao eixo de coordenadas (parte superior da figura) e rotações de um eixo em torno de um ponto (parte inferior da figura) [2].

Finalmente, pode-se comentar sobre o movimento de várias rotações aplicadas sucessivamente em um ponto. No caso de duas rotações, q_1 e q_2, o ponto final será dado por

$$p_{Final} = q_1(q_2 p_0^1 q_2^*)q_1^* = (q_1 q_2)p_0^1(q_2^* q_1^*). \tag{2.74}$$

Nesta equação se observa também que a aplicação sucessiva de transformações pode ser dada pela sua multiplicação.

2.4.4 Quatérnios Duais Unitários

Enquanto é possível utilizar um quatérnio puro para a representação de uma posição no espaço ou translação, e um quatérnio unitário para a representação de uma rotação, não há como mapear o movimento completo de um corpo a partir de um quatérnio. De modo a se realizar uma transformação de translação e rotação simultaneamente, utiliza-se os quatérnios duais unitários, definidos na Sec. 2.3.4.1 como o grupo $Spin(3)$.

Para realizar uma transformação de movimento completo se considera primeiro uma translação, dada pelo quatérnio puro t, seguida por uma rotação, dada pelo quatérnio unitário r. Assim, se tem a definição do quatérnio dual unitário como

$$\underline{x} = r + \varepsilon\frac{tr}{2}. \tag{2.75}$$

A partir de (2.75) pode-se extrair a translação e rotação — o resultado da rotação

$$r = \mathcal{P}(\underline{x}) \tag{2.76}$$

é intuitivo visto que a rotação é definida como parte primária de um quatérnio; além disso, utilizando as definições de parte primária e dual de um quatérnio dual (2.45) e definição da inversa de um quatérnio unitário (2.30), pode-se obter com uma simples manipulação matemática o resultado desenvolvido em

$$\begin{aligned}
\underline{x} =& r + \varepsilon\frac{tr}{2} \\
t =& \frac{2\,(\underline{x} - r)}{r}, \;\; \underline{x} = \mathcal{P}(\underline{x}) + \mathcal{D}(\underline{x}) \\
t =& \frac{2\,(\mathcal{P}(\underline{x}) + \mathcal{D}(\underline{x}) - \mathcal{P}(\underline{x}))}{\mathcal{P}(\underline{x})} \\
t =& 2\mathcal{D}(\underline{x})\mathcal{P}(\underline{x})^{-1}, \;\; \mathcal{P}(\underline{x})^{-1} = \mathcal{P}(\underline{x})^* \\
t =& 2\mathcal{D}(\underline{x})\mathcal{P}(\underline{x})^*.
\end{aligned} \tag{2.77}$$

Além disso, pode-se descrever tanto a translação pura como $p_0 = \begin{bmatrix}1\ 0\ 0\ 0\ 0\ p_x\ p_y\ p_z\end{bmatrix}^T$ quanto quatérnios de rotação pura, que podem ser definidos a partir da forma polar definida em (2.31) em que $r = \left[cos(\frac{\phi}{2})\;\; sen(\frac{\phi}{2})v_1\;\; sen(\frac{\phi}{2})v_2\;\; sen(\frac{\phi}{2})v_3\right]$ e $v = [v_1\ v_2\ v_3]$ é um vetor unitário.

Finalmente, pode-se aplicar o movimento a um ponto (como o centro de massa de um corpo rígido), a partir de transformações semelhantes as definidas na Sec. 2.4.3. Desse modo, definindo um

ponto no espaço como o quatérnio dual p_0, pode-se aplicar uma transformação genérica q, de modo que, em relação a um sistema de coordenadas $\mathcal{F}$, o ponto resultante será dado por p', como em

$$p^{'} = qp_0q^*, \tag{2.78}$$

que é ilustrada pela Fig. 2.5.

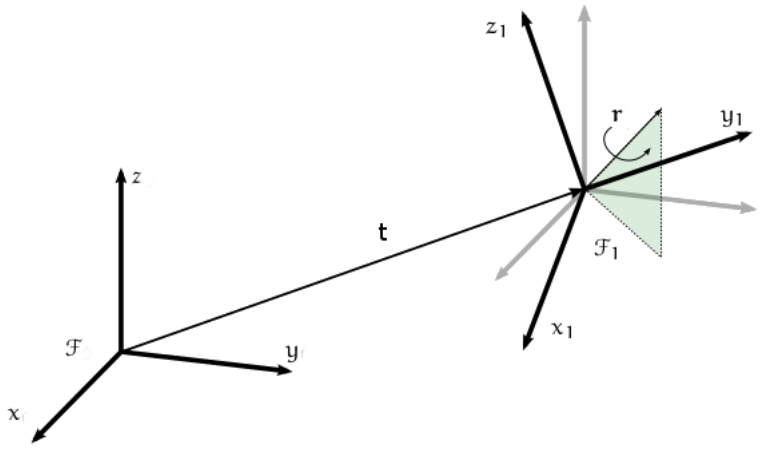


Fig. 2.5: Movimento completo de um ponto, incluindo a translação t e a rotação r [2]

2.5 CINEMÁTICA DE MANIPULADORES

Um manipulador robótico, geralmente caracterizado como um braço, pode ser definido como uma cadeia de corpos rígidos (elos) que são ligados por juntas. As juntas podem ser classificadas entre diversos tipos, em particular entre juntas prismáticas, cujos elos deslizam de maneira linear um sobre o outro, e juntas de revolução, que rotacionam em torno de um eixo definido como o ponto de interseção entre elas. Ambas as ligações são mostradas na Fig.2.6 [3].

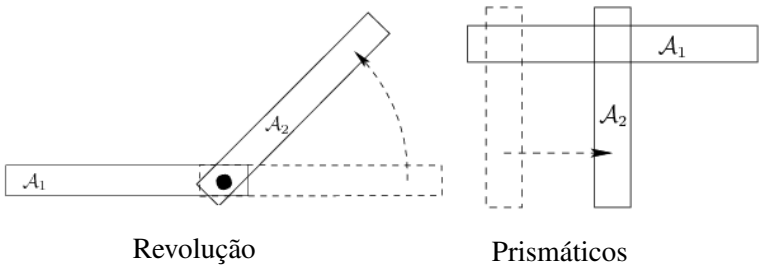


Fig. 2.6: Tipos de Junta [3]

Em analogia ao corpo humano, cada junta teria a função de uma articulação, permitindo mais mobilidade. De fato, na robótica há o conceito de *Grau de Liberdade*[5], que indica o número de

[5]Também referido como DoF, do inglês, *Degree of Freedom*

parâmetros que podem variar independentemente. Se remetendo novamente a Fig.2.6, nota-se que para ambos os tipos de junta, os elos $\mathcal{A}_1$ e $\mathcal{A}_2$ poderão se movimentar somente em uma direção. Desse modo, cada junta prismática ou de revolução concederá um grau de liberdade ao manipulador[6].

Pode-se definir, também, uma divisão entre certas partes do manipulador: o seu "corpo" que será o braço em si, cuja função é garantir mobilidade; um pulso, que deve lhe dar destreza e um efetuador final[7], que realiza a tarefa requerida pelo robô. A Fig. 2.7 mostra esta divisão em um manipulador de um robô humanoide. Em particular, pode-se notar a presença de juntas, neste caso de revolução, ligando os diferentes elos do braço, destaca-se também a possibilidade de uma composição de juntas ligando os mesmos dois elos.

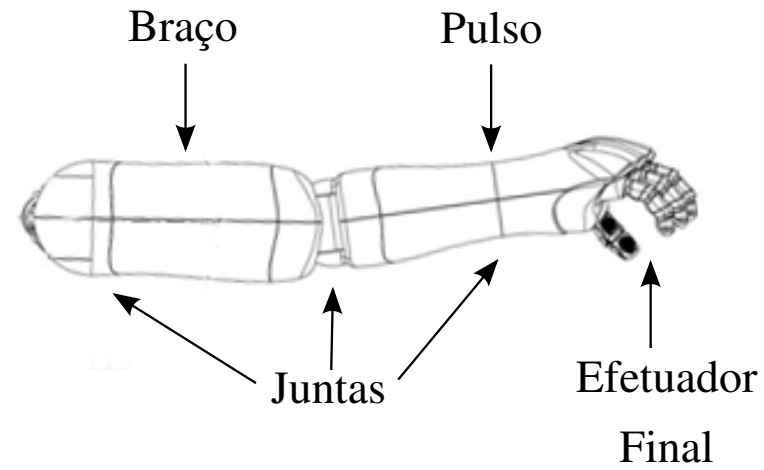


Fig. 2.7: Exemplo de um braço de humanoide *(fonte: http://doc.aldebaran.com/)*

Se remetendo a Sec.2.4, tem-se que um movimento de um corpo rígido em um espaço tridimensional pode ser descrito minimamente através de três parâmetros que denotam rotação e três parâmetros que denotam translação, de modo que se tornam necessários no mínimo seis parâmetros para a descrição completa do movimento — em outras palavras, o movimento no espaço tridimensional é descrito por seis graus de liberdade. Logo, um manipulador precisaria ter no mínimo seis juntas de 1DoF, distribuídas pela sua estrutura, de modo a alcançar qualquer posição no espaço. Manipuladores com mais do que seis juntas passam a ser redundantes, possuindo variáveis de atuação extras.

Finalmente, tem-se o conceito de espaço de trabalho[8], que dependerá da estrutura do manipulador, do número de juntas e sua disposição. O espaço de trabalho será definido como o espaço que o efetuador final poderá atuar.

2.5.1 Parâmetros de Denavit-Hartenberg

Criados por Jacques Denavit e Richard Hartenberg em 1955 [33], os parâmetros de Denavit-Hartenberg, também designados como parâmetros DH, são quatro indicadores que caracterizam a posição relativa entre dois eixos de coordenadas. Tal notação é utilizada na robótica, de modo geral, para definir a relação entre eixos de coordenadas ligados a diferentes elos de uma cadeia cinemática.

[6]Existem outros tipos de junta com mais graus de liberdade, um exemplo seria a junta esférica que possui 3DoF.
[7]Muitas vezes referido por seu termo em inglês, *End Effector*
[8]Do inglês, *Workspace*

De fato, existem diversas outras padronizações para representar a construção física de um manipulador, entretanto esta é a mais difundida na comunidade robótica.

Para se obter os quatro parâmetros, denominados a, α, d e θ, deve-se seguir uma série de diretivas de forma a definir dois eixos de coordenadas em duas juntas, i e $i+1$, e então determinar a relação entre elas. Para definir os eixos de coordenadas [34]:

1. Escolhe-se o eixo z_i de forma a alinhá-lo com o eixo de movimentação da junta, isto é, eixo de rotação para juntas de revolução ou eixo de translação para juntas prismáticas.
2. O eixo x_i deve ser posicionado paralelo à normal comum[9] entre z_i e z_{i-1}. Caso i seja a primeira junta, a posição de x_i é arbitrária. Deve-se notar também que, caso z_{i-1} e z_i sejam paralelos, existem infinitas normais comuns entre eles, portanto pode-se escolher dentre elas a que mais se adequar à aplicação.
3. O eixo y_i é definido de acordo com x_i e z_i, utilizando-se a Regra de Fleming[10].

Pode-se então definir os parâmetros DH:

1. a: Comprimento da normal comum.
2. α: Ângulo entre z_{i-1} e z_i em torno da normal comum.
3. d: Deslocamento ao longo do eixo z_{i-1} até a normal comum.
4. θ: Ângulo entre x_{i-1} e x_i em torno do eixo z_{i-1}.

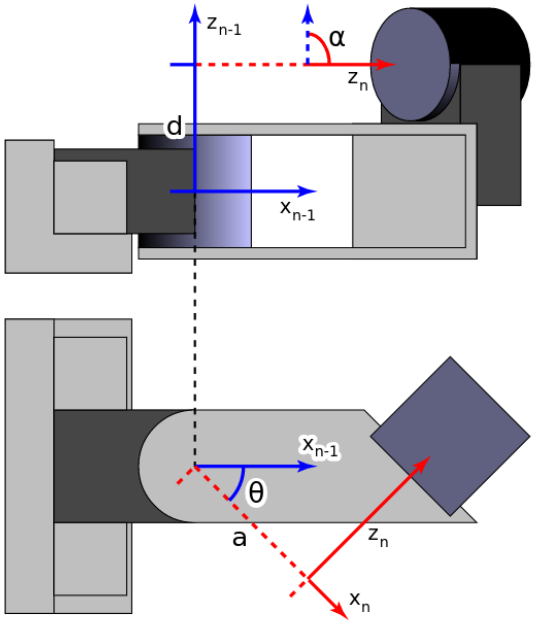


Fig. 2.8: Parâmetros de Denavit-Hartenberg (*fonte: http://wiki.tekkotsu.org/*)

[9] Segmento de reta que caracteriza a distância mínima entre duas linhas.
[10] Popularmente conhecida como regra da mão direita.

Obtidos os parâmetros, pode-se calcular a transformação entre duas juntas da cadeia cinemática aplicando operações de translação e de rotação. Utilizando-se matrizes homogêneas, tem-se [35]

$$T_{i-1}^{i} = T_{DH} = T_{rot,z}(\theta)T_{trans,z}(d)T_{trans,x}(a)T_{rot,x}(\alpha) = \begin{bmatrix} c\theta & -s\theta.c\alpha & s\theta.s\alpha & a.c\theta \\ s\theta & c\theta.c\alpha & -c\theta.s\alpha & a.s\theta \\ 0 & s\alpha & c\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.79}$$

em que $T_{rot,a}$ é uma rotação em torno do eixo a e $T_{trans,a}$ é uma translação ao longo do eixo a.

De forma análoga, se utilizando quatérnios duais, tem-se [6, 2]

$$q_{i-1}^{i} = q_{DH} = q_{rot,z}(\theta)q_{trans,z}(d)q_{trans,x}(a)q_{rot,x}(\alpha) \tag{2.80}$$

em que:

$$\begin{aligned} q_{rot,z}(\theta) =& cos\frac{\theta}{2} + sen\frac{\theta}{2}\hat{k} \\ q_{trans,z}(d) =& 1 + \left(\frac{1}{2}d\right)\hat{k}\varepsilon \\ q_{trans,x}(a) =& 1 + \left(\frac{1}{2}a\right)\hat{i}\varepsilon \\ q_{rot,x}(\alpha) =& cos\frac{\alpha}{2} + sen\frac{\alpha}{2}\hat{i}. \end{aligned}$$

Para uma cadeia cinemática de n juntas, a transformação da base até o efetuador final pode ser dada por

$$T_0^n = T_0^1 T_1^2 \dots T_{n-1}^n, \tag{2.81}$$

utilizando matrizes homogêneas, e

$$q_0^n = q_0^1 q_1^2 \dots q_{n-1}^n \tag{2.82}$$

utilizando-se quatérnios duais [2].

2.5.2 Cinemática Direta

A cinemática direta define o mapeamento da configuração de juntas do robô para a posição e orientação do efetuador final. Dado um manipulador com n juntas $(\theta_1, \theta_2, \dots, \theta_n)$, a função de cinemática direta f pode ser definida como em

$$x = f(\theta). \tag{2.83}$$

Em que x representa a posição e orientação do efetuador e θ é o vetor de juntas. A solução de tal problema é trivial depois de obtidos os parâmetros DH do manipulador, logo, f pode ser representada

por (2.81) ou (2.82) utilizando-se a representação matricial ou de quatérnios duais, respectivamente. Note que a cinemática direta é uma função que sempre possui uma solução analítica.

2.5.3 Cinemática Inversa

Manipuladores tipicamente precisam realizar trajetórias e alcançar pontos no espaço tridimensional. Para atingir tais posições deve-se calcular os valores de cada junta do manipulador. A cinemática inversa define a relação e não o mapeamento como na cinemática direta, entre uma posição e orientação no espaço e os valores de juntas do robô. De foma análoga à (2.83), a cinemática inversa pode ser definida como

$$\theta = f^{-1}(x). \tag{2.84}$$

De acordo com [1], a solução desta função pode possuir um nível de complexidade maior do que aquele encontrado na cinemática direta, pois:

1. As equações a serem resolvidas são geralmente não lineares, logo uma solução fechada pode nem sempre existir.
2. Podem existir múltiplas soluções.
3. Podem existir soluções infinitas, como por exemplo no caso de manipuladores redundantes.
4. Uma solução admissível pode não existir, dado os limites estruturais do manipulador.

2.5.4 Cinemática Diferencial

A cinemática definida até aqui se baseia na relação entre uma posição no espaço e a configuração de juntas de um manipulador. A cinemática diferencial, por outro lado, visa estabelecer a relação entre a velocidade do efetuador final e a velocidade das juntas do manipulador. Esta relação pode ser obtida aplicando a diferenciação de (2.83) o que leva à

$$\dot{x}(t) = J(\theta)\dot{\theta}(t) \tag{2.85}$$

tal que

$$J(\theta) = \frac{\delta f_i(\theta)}{\delta \theta_j}$$

Considerando um sistema em que $\dot{x}(t)$ possui m dimensões e $\dot{\theta}(t)$ possui n, $J(\theta)$ pode ser definido como mostrado em

$$J(\theta) = \begin{bmatrix} \frac{\delta f_1(\theta)}{\delta \theta_1} & \cdots & \frac{\delta f_1(\theta)}{\delta \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_m(\theta)}{\delta \theta_1} & \cdots & \frac{\delta f_m(\theta)}{\delta \theta_n} \end{bmatrix}. \tag{2.86}$$

De forma análoga à (2.84), a cinemática inversa diferencial pode ser definida na forma de

$$\dot{\theta}(t) = J^{-1}(\theta)\dot{x}(t). \tag{2.87}$$

Dado que a matriz J nem sempre será uma matriz quadrada, em geral utiliza-se a sua pseudoinversa como mostrado em

$$\dot{\theta}(t) = J^{\dagger}(\theta)\dot{x}(t), \tag{2.88}$$

no qual $J^{\dagger}(\theta)$ é a pseudoinversa pela direita, definida da forma:

$$J^{\dagger} = J^{T}\left(JJ^{T}\right)^{-1}. \tag{2.89}$$

A matriz Jacobiana possui diversas informações importantes sobre a estrutura de um manipulador. O número de linhas linearmente independentes é equivalente ao número de graus de liberdade controláveis. O número de colunas é igual ao número de graus de liberdade no espaço das juntas. Considerando ainda J como uma matriz de dimensões $(m \times n)$, se n for menor que m temos que o manipulador é sub-atuado, caso n for igual a m e J for de *posto* completo tem-se uma configuração suficiente para atingir os objetivos no espaço cartesiano, e se n for maior que m o manipulador é redundante.

Pode-se verificar também que, caso o manipulador se encontre em uma configuração singular, a Jacobiana perde um ou mais postos, deixando de ser posto completo e perdendo a inversibilidade. Configurações singulares podem ser facilmente verificadas calculando seu determinante. Caso $det(J(\vec{\theta})) = 0$ [1], a Jacobiana é dita singular, caso a Jacobiana não seja quadrada pode-se checar $det\left(JJ^{T}\right) = 0$ de modo a obter a mesma informação. Assim, a partir desta configuração singular, existirão direções ao longo das quais é impossível movimentar o efetuador, independentemente da velocidade das juntas. Se não tratadas adequadamente, ao se obter a cinemática inversa em posições singulares tem-se como resultado velocidades de junta elevadas (em teoria, infinitas), que podem sobrecarregar os atuadores. Em uma aplicação real, ao se aproximar de singularidades, o manipulador apresenta sinais de vibrações nas juntas e instabilidade.

Visando atenuar os efeitos de singularidades, pode-se utilizar a pseudoinversa amortecida. Tal solução numérica adiciona um fator de amortecimento λ não nulo, como mostrado em

$$J^{\dagger} = J^{T}\left(JJ^{T} + \lambda^{2}I\right)^{-1}. \tag{2.90}$$

Note que se utiliza λ^2 de forma a garantir que seu sinal seja positivo.

Assim, pode-se garantir que os autovalores da matriz $JJ^{T} + \lambda^{2}I$ serão sempre positivos, inclusive em configurações singulares. Não obstante, o fator de amortecimento adiciona erros no mapeamento de velocidades que resultam em erros no transitório da trajetória [36].

2.6 CONTROLE CINEMÁTICO DE MANIPULADORES

Um sistema de controle pode ser definido como uma composição de componentes que juntos condicionarão uma variável de entrada de modo a se obter uma saída desejada. De forma mais simples, tal processo pode ser entendido pela Fig.2.9, em que dado um processo qualquer, é possível obter uma saída[11] [4].

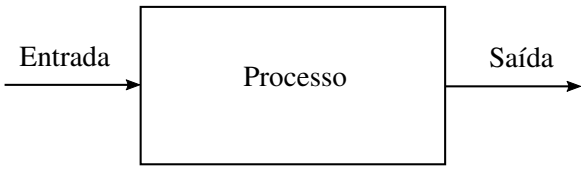


Fig. 2.9: Representação do processo básico de controle, *adaptado de* [4]

A arquitetura mais básico de controle se dá por meio da comparação entre a entrada e a saída de um sistema, ou seja, a partir do sensoriamento da variável de entrada é realizado um *feedback* negativo em que se obtêm um erro. O controlador deverá então agir de modo a minimizar este erro e gerar um sistema estável. Assim a Fig.2.10 mostra um esquema genérico da estrutura de um controlador.

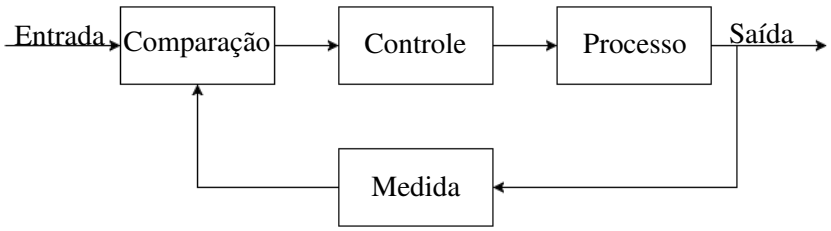


Fig. 2.10: Representação de controlador com realimentação, *adaptado de* [4]

Existem diversas técnicas para se definir o controlador, assim como diferentes tipos. É bastante comum se utilizar controladores proporcionais, integrais ou derivativos, em que as respectivas operações são utilizadas para dimensionar a ação de controle em relação ao erro.

Para manipuladores é comum se realizar o controle cinemático, em que as variáveis de controle estão relacionadas a posições, velocidades ou grandezas derivadas a partir destas. Aqui o foco estará no controle de posição e de orientação de um manipulador no espaço tridimensional, particularmente aplicados ao efetuador final do robô.

Tomando um vetor com todas as posições de juntas de um manipulador, dado por θ, pode-se utilizar a cinemática direta (Sec. 2.5.2) com a descrição dos parâmetros de Denavit-Hartenberg referentes ao robô utilizado (Sec.2.5.1) para se obter a posição do seu efetuador final, aqui denotado por x. Desse modo, utilizando as Jacobianas definidas em (2.87) ou (2.88) pode-se obter o vetor de velocidades $\dot{\theta}$. Assim o controle poderá ser realizado a partir da integração das velocidades no tempo, como em

[11] A representação do controlador dada pela Fig.2.9 é a de uma caixa preta. Dentro do processo são definidos diversos subprocessos que se comunicam de diferentes maneiras, podendo haver relações de *feedback*, soma, etc.

$$\theta(t) = \int_0^t \dot{\theta}(\varsigma) d\varsigma + \theta(0), \tag{2.91}$$

em que $\theta(t)$ será a posição a qual se deseja controlar e $\theta(0)$ a condição inicial [1].

Para realizar a integração em tempo discreto, de modo a se aplicar o processo em sistemas digitais, pode-se utilizar o método de integração de Euler. Assim obtêm-se

$$\theta\left(t_{k+1}\right) = \theta\left(t_k\right) + \dot{\theta}\left(t_k\right) \triangle t, \tag{2.92}$$

em que Δt é o intervalo de integração e $t_{k+1} = t_k + \Delta t$.

Desse modo, pode-se definir a equação para um método de controle proporcional de um sistema como na

$$\dot{\theta}_{k+1} = J^{-1}\left(\dot{\theta}_k\right)\ \left(x_{k+1} + K x_e\right). \tag{2.93}$$

Em que K é uma matriz de ganhos e $\overrightarrow{x_e}$ é o vetor de erros do sistema.

Assim como se deseja que o sistema se estabilize em uma posição e pare, coloca-se que a velocidade no tempo $k+1$ será $x_{k+1} = 0$, de modo que a lei de controle deverá ser derivada a partir da (2.92) e modo a ser obter

$$\theta_{k+1} = \theta_k + J^{-1} K x_e. \tag{2.94}$$

Finalmente, dado que a Jacobiana nem sempre é definida como uma matriz quadrada, tem-se que a lei de controle pode utilizar a pseudoinversa da Jacobiana ou a pseudoinversa amortecida, como definido na Sec.2.5.4. Assim define-se

$$\theta_{k+1} = \theta_k + J^{\dagger} K x_e. \tag{2.95}$$

Finalmente, na Fig.2.11 é mostrado o diagrama de blocos referente ao sistema de controle cinemático para um manipulador, como definido em (2.95).

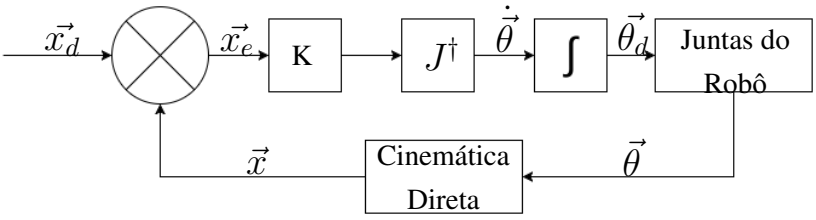


Fig. 2.11: Diagrama de blocos representando o controle cinemático de um manipulador, *adaptado de [1]*

2.6.1 Espaço Nulo

Considerando um manipulador redundante tem-se que o número de DoFs é superior ao número de variáveis a serem controladas, assim dependendo da tarefa a ser controlada podem sobrar graus de liberdade e desse modo há um ou mais graus extras os quais podem ser utilizados para executar tarefas secundárias. Para tanto define-se o conceito de espaço nulo. Este é um subespaço no domínio da velocidade das juntas que quando mapeado pela Jacobiana não produz alterações na velocidade do efetuador final. Note que tal espaço pode ser utilizado em manipuladores não redundantes ou subatuados desde que a tarefa a ser executada possua um número de variáveis menor que o número de DoFs do manipulador. A Fig.2.12 mostra como o espaço nulo é mapeado para o domínio da velocidade do efetuador.

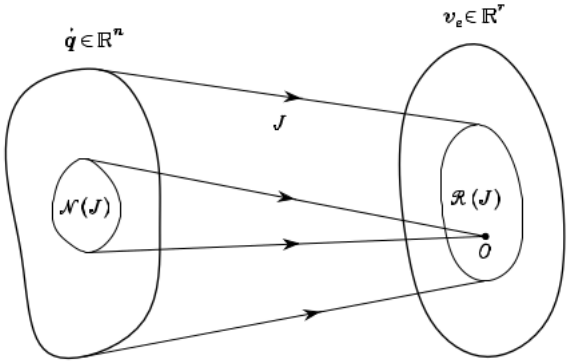


Fig. 2.12: Mapeamento do Espaço Nulo a partir da Jacobiana *[1]*

Matematicamente o espaço nulo pode ser definido como em

$$\eta = \left(I - J^{\dagger}J\right)\dot{\theta}_0, \tag{2.96}$$

no qual I é a matriz identidade de dimensão $(n \times n)$, define-se também a parcela $P_i = \left(I - J^{\dagger}J\right)$ como o projetor do espaço nulo. É fácil verificar que $JP_i = 0$ para qualquer J.

Nota-se também que $\dot{\theta}_0$ é um vetor de velocidades de juntas arbitrárias a ser definido pela tarefa secundária e que não afetará a tarefa principal pois $J\eta = 0$.

2.7 CONTROLE NO ESPAÇO COOPERATIVO

O problema de se realizar tarefas em que mais de um manipulador pudesse cooperar surgiu com o intuito de melhorar tarefas que um braço sozinho não pudesse realizar. Em 1970 já podiam ser vistos exemplos de soluções multiagentes, por exemplo com controle mestre-escravo, controle de força/complacência ou a definição do controle no espaço de tarefa [8]. De fato, nos últimos anos diversas pesquisas vêm sendo feitas de modo a buscar soluções eficientes para a cooperação entre robôs, visto que a complexidade da tarefa e do controle aumenta significativamente.

Visando solucionar o problema de da cooperação, definiu-se um método de controle voltado para o espaço de tarefas — o controle no espaço cooperativo. Esta técnica define que o controle deverá ser

feito em termos do movimento relativo e absoluto do sistema cooperativo, computados diretamente a partir do efetuador final dos manipuladores.

Visto que se caracteriza o controle no espaço cooperativo em termos de movimentos relativos e absolutos, é possível, dada a pose x_1 e x_2 de dois efetuadores, equacionar pose absoluta como o ponto médio entre eles, dada por

$$x_{abs} = \frac{x_1 + x_2}{2}. \tag{2.97}$$

Além disso, a pose relativa é definida como a distância entre ambos, o que é dado em

$$x_{rel} = x_2 - x_1. \tag{2.98}$$

Somente a pose absoluta não seria capaz de representar o sistema cooperativo por completo. É necessário também que seja definido a pose de um manipulador em relação ao outro, afinal o ponto médio entre eles poderá ser o mesmo para várias distâncias relativas, o que torna indispensável a definição da distância relativa.

Tais variáveis podem ser, portanto, utilizadas tanto para a manipulação de objetos, inclusive objetos não rígidos, quanto para a descrição de movimentos coordenados em que não haverá um ponto físico em comum entre os braços e desse modo não será possível realizar um controle a partir dos esforços realizados.

2.7.1 Espaço Cooperativo Dual

A definição de espaço cooperativo pode ser expandida para a representação de movimento com quatérnios duais levando ao conceito de espaço cooperativo dual [2, 37, 6]. Assim, aqui será utilizada a posição dos efetuadores finais dadas pelos quatérnios duais $\underline{x}_1$ e $\underline{x}_2$, de modo que a pose absoluta ainda seja definida como o ponto médio entre ambos os efetuadores e a relativa como a distância entre eles, a Fig.2.13 exemplifica como é esta relação no espaço.

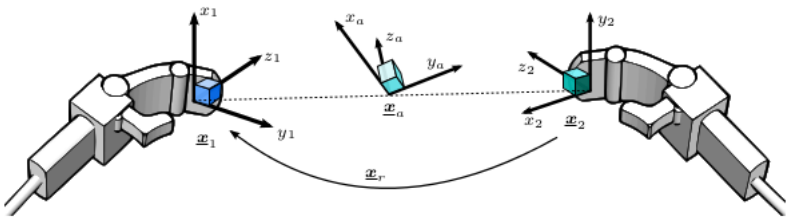


Fig. 2.13: Representação do Espaço Cooperativo entre dois manipuladores [2]

Desse modo, a pose relativa do braço primeiro braço em relação ao segundo será dada por

$$\underline{x}_{rel} = \underline{x}_2^* \underline{x}_1 \tag{2.99}$$

e a absoluta por

$$\underline{x}_{abs} = \underline{x}_2 \underline{x}_{rel/2} \tag{2.100}$$

em que $\underline{x}_{rel/2}$ é definido como metade do ângulo de rotação da parte primária do quatérnio dual unitário $\underline{x}_{rel}$. Pode-se definir também $\underline{x}_{rel/2}$ como

$$\underline{x}_{rel/2} = (\underline{x}_{rel})^{1/2} = exp\left(\frac{log\,(\underline{x}_{rel})}{2}\right). \tag{2.101}$$

Finalmente, dadas as definições das variáveis do espaço cooperativo e de controle (Sec.2.6), é possível definir um vetor de juntas aumentado, compostos pela concatenação dos vetores das juntas de cada braço, dado por

$$\Theta = \left[\theta_1^T\ \theta_2^T\right]^T \tag{2.102}$$

Desse modo o controle poderá ser efetuado por meio de Jacobianas aumentadas, calculadas a partir das definições de (2.99) e (2.100).

3 FERRAMENTAS DE DESENVOLVIMENTO

"Everything must be
made as simple as possible. But not simpler."

Albert Einstein

3.1 INTRODUÇÃO

Para a realização do projeto apresentado neste manuscrito foram usadas diversas ferramentas, que serão apresentadas neste capítulo. Primeiramente será feita uma descrição mais detalhada da plataforma NAO, em seguida, serão apresentadas as ferramentas computacionais que foram utilizadas: o simulador V-Rep, a plataforma ROS, e o pacote de DQ Robotics.

3.2 PLATAFORMA NAO

Para este projeto foi utilizada a plataforma NAO, produzida pela Aldebaran Robotics [5]. Este pequeno humanoide se popularizou pela sua aparência cativante e pela vasta gama de aplicações em que ele vem sendo utilizado, notavelmente interações com crianças autistas, interação humano-robô, controle remoto por teleoperação, além, é claro, da sua participação na categoria de plataforma padrão da RoboCup.

Em termos de Hardware, o NAO é construído com um processador Intel Atom de 1.6 GHz e 1 GB de RAM, possuindo também uma vasta gama de sensores, incluindo dois pares receptor/transmissor de sonares, a unidade inercial, que é composta por um giroscópio de dois eixos e um acelerômetro de três, localizados no torso. Além destes, o NAO possuí também dois sensores infravermelho nos olhos, quatro sensores de pressão em cada pé, sensores capacitivos na cabeça e em cada mão, um botão no torso, um *bumper* em cada pé e encoders que fornecem a posição de cada junta. Finalmente o robô também possui duas câmeras, com resolução de 640x480 pixeis, uma localizada na parte frontal superior da cabeça e a outra na parte frontal inferior. A Fig. 3.1 mostra os sensores presentes no robô.

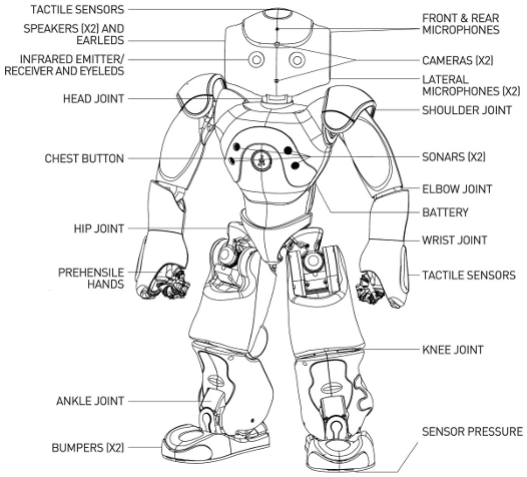
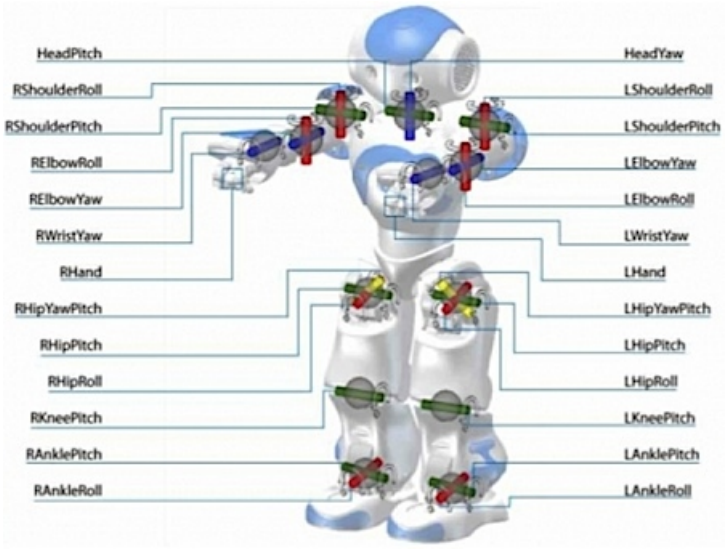
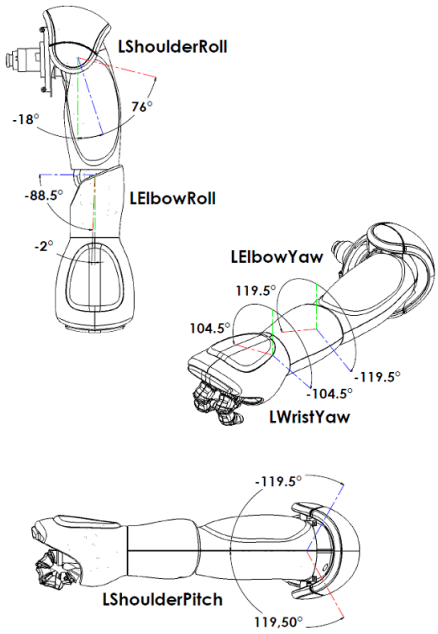


Fig. 3.1: Esquemático das posições dos principais sensores do NAO [5]

Ainda no escopo da estrutura física do robô, é importante enfatizar que este possui diversos atuadores, como mostrado na Fig. 3.2a. Ao todo o NAO conta com 25 graus de liberdade, mais especificamente 5 em cada um de seus braços (Fig. 3.2b). Por fim, as principais medidas dos elos do robô são mostradas na Fig. 3.3 e também, resumidas àquelas necessárias para a descrição dos elos de um braço, na Tab. 3.1.



(a)

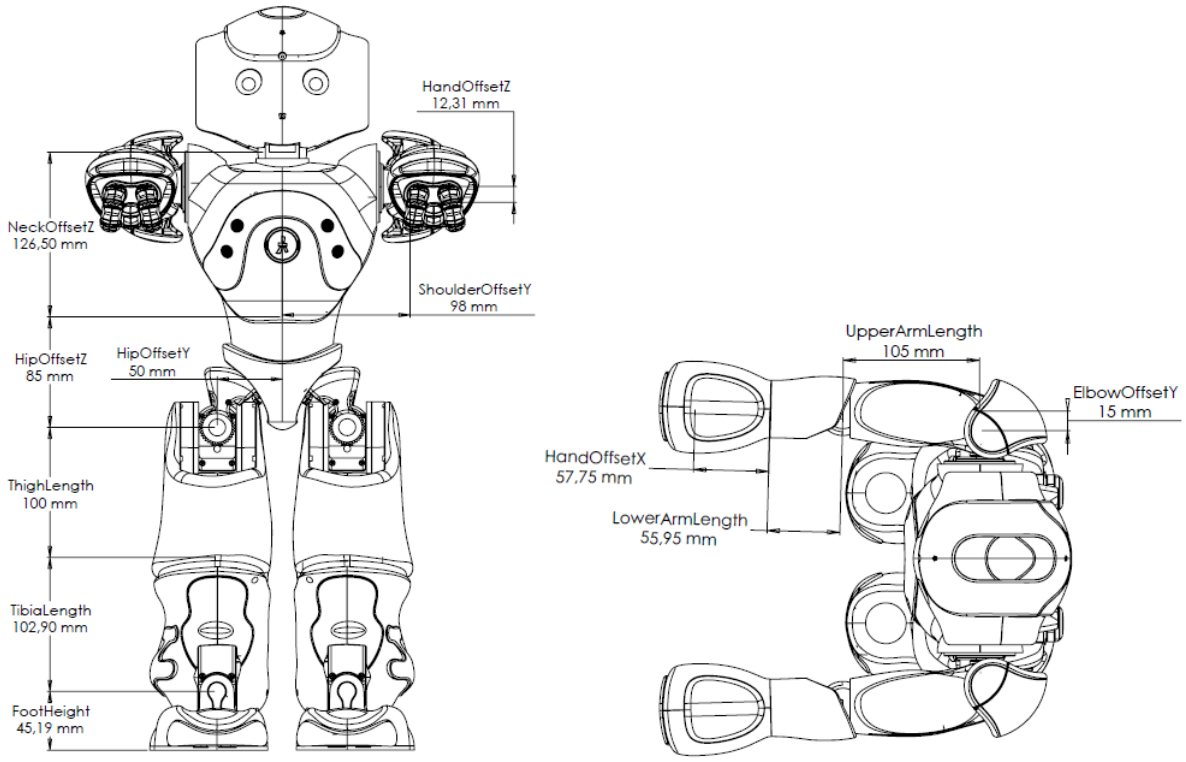


(b)

Fig. 3.2: Esquemático das posições dos principais atuadores do NAO [5]

Tab. 3.1: Comprimento dos elos relacionados aos braços do robô.

Elo	Comprimento (mm)
ShoulderOffsetY	98.00
ShoulderOffsetZ	100.00
ElbowOffsetY	15.00
UpperArmLength	105.00
LowerArmLength	55.95
HandOffsetX	57.75
HandOffsetZ	12.31



(a) Vista Frontal

(b) Vista Superior

Fig. 3.3: Comprimento dos elos principais do robô. [5]

Em termos de software, o robô opera utilizando um sistema operacional baseado na distribuição do Linux, Gentoo. O acesso a memória para a obtenção dos dados dos sensores ou a atuação sobre as suas juntas são feitos através de um framework disponibilizado pela Aldebaran Robotics, o NAOqi. Este framework chama os diversos módulos disponíveis através da criação de um Broker que carrega as bibliotecas referentes aos diferentes métodos que se deseja utilizar. A Fig. 3.4 faz referência a esse processo.

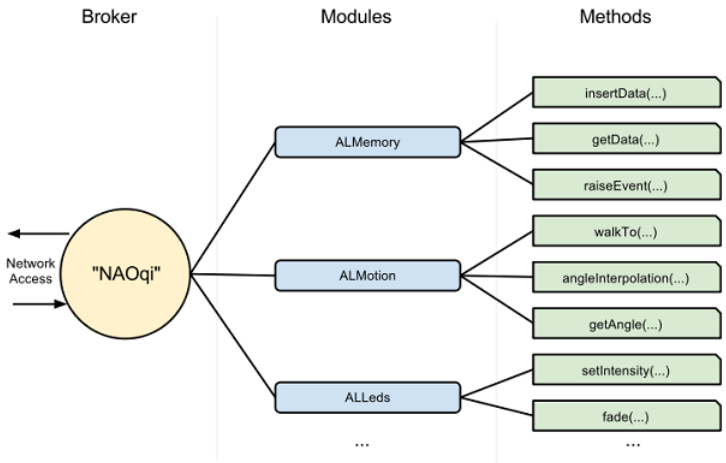


Fig. 3.4: Esquemático da estrutura do Sistema disponibilizado pela NAOqi [5]

3.3 OPENCV

Ao iniciar um projeto de visão computacional, deve-se escolher dentre as ferramentas disponíveis no mercado aquela que melhor se aplica às restrições do seu projeto, tais como sistema operacional, linguagem de programação a ser utilizada e poder computacional da plataforma disponível.

O OpenCV (Open Source Computer Vision) é uma biblioteca multiplataforma, de código aberto, desenvolvido inicialmente pela Intel Corporation. A biblioteca possui mais de 500 funções, divididas entre módulos de estrutura de dados, álgebra linear, entrada e saída de vídeos, interface com o usuário, processamento de imagens e diversos algoritmos de visão computacional, dentre eles, calibração de câmera, filtros de imagem, reconhecimento de objetos e pessoas, etc. É importante notar que o seu processamento é em tempo real, permitindo que possa ser utilizado em aplicações online como, por exemplo, a robótica.

O OpenCV é desenvolvido principalmente nas linguagens de programação C/C++, entretanto, também é oferecido suporte para as linguagens Java, Python e Visual Basic.

3.4 AMBIENTE DE SIMULAÇÃO V-REP

Quando se trata de robótica (além de diversos outros segmentos da engenharia), uma prática muito comum é se utilizar simuladores para validar experimentos e aplicações com segurança, sendo somente após o sucesso em simulação que os sistemas validados são embarcados na plataforma física a ser utilizada. Visando realizar tal validação, o presente trabalho escolheu a plataforma V-Rep (Virtual Robotics Experimentation Platform)[1] como ambiente de simulação (Fig. 3.5).

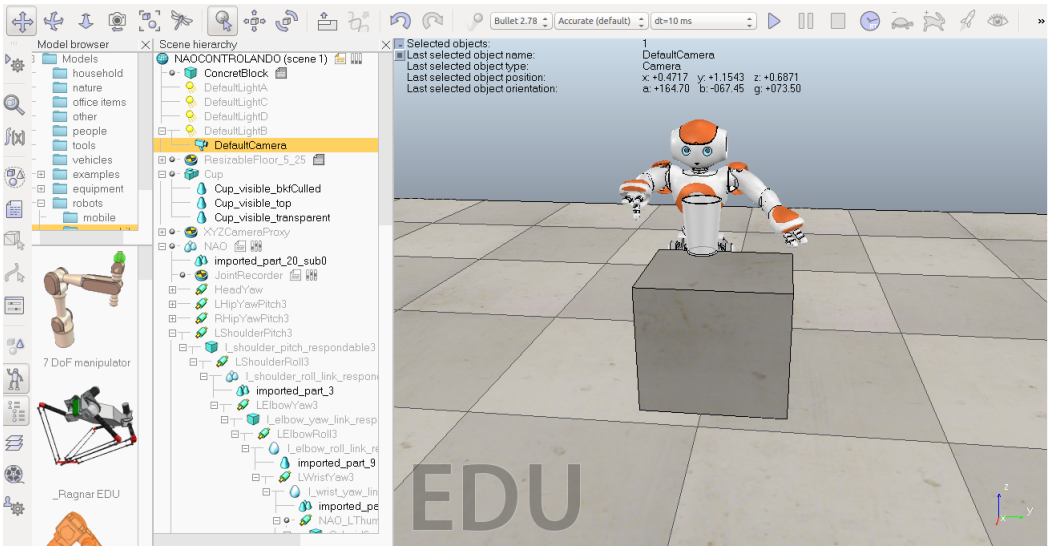


Fig. 3.5: V-Rep

O V-Rep é um software comercial (com versão estudantil gratuita) para plataformas robóticas desenvolvido pela Coppelia Robotics. Este simulador tem o intuito de ser uma plataforma versátil,

[1] http://www.coppeliarobotics.com/

em que uma grande gama de linguagens de programação pode ser usadas para a implementação de modelos e controladores e em que códigos são facilmente portáveis para as plataformas físicas. O funcionamento do simulador se dá a partir da criação de cenas contendo diversos objetos que são integrados a partir de uma hierarquia em forma de árvore, de modo que módulos como de dinâmica, cinemática, planejamento de trjetória, etc. são utilizados para ditar as leis de interação entre os objetos. [38]

3.5 ROS

A robótica conta com uma grande comunidade de pesquisadores, um fato que torna possível a construção de novas tecnologias cada vez mais complexas. Entretanto, diferentes robôs possuem diferentes arquiteturas, o que geralmente dificulta a importação de códigos entre plataformas e utilização de algumas ferramentas com certos sistemas. Com o intuito de simplificar o desenvolvimento e possibilitar uma maior integração em termos de software, diversas ferramentas foram desenvolvidas, por exemplo, o framework ROS (Robotics Operating System)[2]. [39]

O ROS é um sistema *open source* baseado em princípios de comunicação de ponto a ponto, com suporte para diversas linguagens de programação, construção modular de suas bibliotecas (o que auxilia a fácil integração com outros sistemas). A sua implementação se dá com base em nós, que são processos modulares, ou seja, elementos da arquitetura do sistema realizando alguma função especifica. Os nós se comunicam através da publicação e subscrição de mensagens ou através da requisição e resposta de serviços entre nós.

Desse modo, para o desenvolvimento de controladores para robótica, o ROS se torna uma ferramenta poderosa, possibilitando a rápida troca de informações, principalmente dada a necessidade de diferentes módulos, tanto para receber dados como atuar sobre eles.

3.6 DQ ROBOTICS

Para se implementar controladores baseados na álgebra dos quatérnios duais em um sistema real, se faz necessário a implementação de algoritmos que definem as operações básicas necessárias. Com esse intuito, foi criada a biblioteca de código aberto DQ Robotics[3]. Tal biblioteca fornece algoritmos envolvendo álgebra de quatérnios duais e cinemática de corpos rígidos e manipuladores. Possui implementações em Matlab, C++ e Python além de pacotes de integração com ROS Indigo e interface com a plataforma V-Rep.

[2] http://www.ros.org/

[3] http://dqrobotics.sourceforge.net/

4 DESENVOLVIMENTO: COMUNICAÇÃO VISUAL

"The world only makes
sense when you force it to"

Batman

4.1 INTRODUÇÃO

Nesta seção será feita a descrição dos métodos desenvolvidos com o intuito de solucionar o desafio "No Wifi" descrito na Sec. 1.1.1.1.

Desse modo observa-se, na Figura 4.1 que os robôs são posicionados um frente ao outro e que para realizar a comunicação visual é necessário a sua detecção. Além disso, para o desafio é necessária também a codificação seguida pela transmissão de uma mensagem.

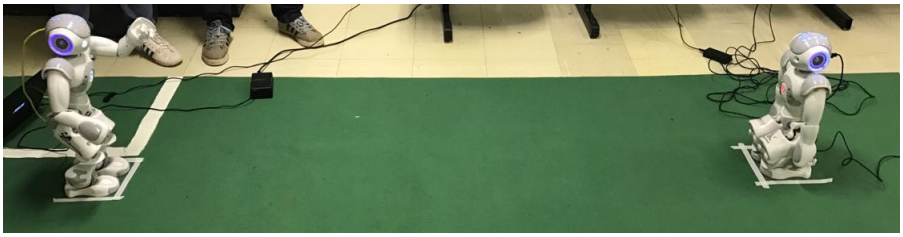


Fig. 4.1: Posicionamento dos robôs para testes

4.2 CODIFICAÇÃO DA MENSAGEM

Em uma partida real o número de interferências externas gerados tanto por outros robôs ou pela torcida inviabilizaria a utilização de comunicação sonora, por isso, optou-se por descartar esta opção. Dentre as formas de comunicação visual, as escolhidas para testes iniciais foram angulação dos braços e cores dos LEDs. Na primeira, dividiu-se o espaço de trabalho de cada braço em uma quantidade finita de posições, e assim cada posição representaria um dígito, enquanto na segunda opção utilizou-se os LEDs encontrados nos olhos e no peito do robô, onde diferentes cores representariam diferentes valores. Percebeu-se que a velocidade de movimentação das juntas do robô tornaria inviável a utilização dos braços, dado que, dentre as regras do desafio, estipulava-se um tempo máximo de comunicação de 15 segundos. Portanto, a forma de comunicação escolhida se dava por meio do uso de visão computacional em que três LEDs poderiam assumir diferentes cores para enviar diferentes mensagens.

Cada LED do robô pode assumir 7 cores diferentes (Amarelo, Azul, Branco, Ciano, Verde, Vermelho e Violeta) além do estado desligado, entretanto, optou-se por utilizar apenas três valores para

a transmissão. Tal escolha foi motivada para garantir que as cores utilizadas pudessem ter o maior contraste entre si, ao mesmo tempo em que ainda seria possível garantir que cada LED pudesse representar pelo menos um bit, ou seja, ainda seria possível transmitir informação utilizando-se a base binaria. Assim, a primeira cor representaria o valor 0, a segunda o valor 1 e a terceira seria responsável por informar a transição entre a transmissão de duas informações diferentes em uma comunicação serial. Uma vez que o NAO possui apenas 3 LEDs frontais, o maior número que pode ser representado em base binária é 7, dessa forma, a base escolhida foi a Octal, cujos dígitos são valores de 0 a 7.

Tab. 4.1: Intervalo de valores assumidos por x e y em decimal e em octal. Os primeiros valores possuem sua origem no centro de campo. Os valores transladados são equivalentes ao tamanho total do campo. Valores seção são os valores máximos que x e y podem assumir dentro de uma seção.

	Valores Originais		Valores Transladados		Valores Seção	
	Decimal	Octal	Decimal	Octal	Decimal	Octal
x mínimo	-4500	-10624_8	0	0_8	0	0_8
x máximo	4500	10624_8	9000	21450_8	2250	4312_8
y mínimo	-3000	-5670_8	0	0_8	0	0_8
y máximo	3000	5670_8	6000	13560_8	3000	5670_8

A mensagem original é composta por dois inteiros representando uma posição cartesiana em milímetros, a qual é equivalente a uma posição real no campo. Considera-se ainda o centro do círculo central como a origem do plano. Deste modo,x e y podem assumir valores nos intervalos mostrados na primeira coluna da Tab. 4.1. Com o propósito de eliminar a necessidade de se representar o sinal da posição, optou-se por mover a origem para um dos vértices do campo, de forma a mapear os valores para o domínio dos números naturais.

Considerando os novos valores máximos para x e y convertidos para a base octal, mostrados na segunda coluna da Tab. 4.1, seriam necessários 10 dígitos para transmitir a mensagem. Dado o tempo limitado para a transmissão da mensagem, fez-se necessário a codificação dos dados de forma a minimizar o tempo de envio. Com o objetivo de reduzir 1 dígito em cada número, dividiu-se o campo em 8 seções diferentes como mostrado na Fig. 4.2, onde cada seção teria sua própria origem e valores máximos de x e y apresentados na terceira coluna da Tab. 4.1. Tomando como base os novos valores convertidos para a base octal, reduziu-se o número máximo de dígitos a serem transmitidos para 9, sendo um deles utilizado para identificar a seção, outros 4 para representar o valor de x e os últimos 4 o valor de y.

Após a codificação dos dados deve-se então serializá-los para que os dígitos possam ser transmitidos via comunicação sem estrutura de redes. Cada dígito na base octal é então convertido para binário, de forma a ser representado pelos três LEDs do robô.

Os passos necessários para codificar e decodificar a mensagem estão representados no diagrama mostrado na Fig. 4.3.

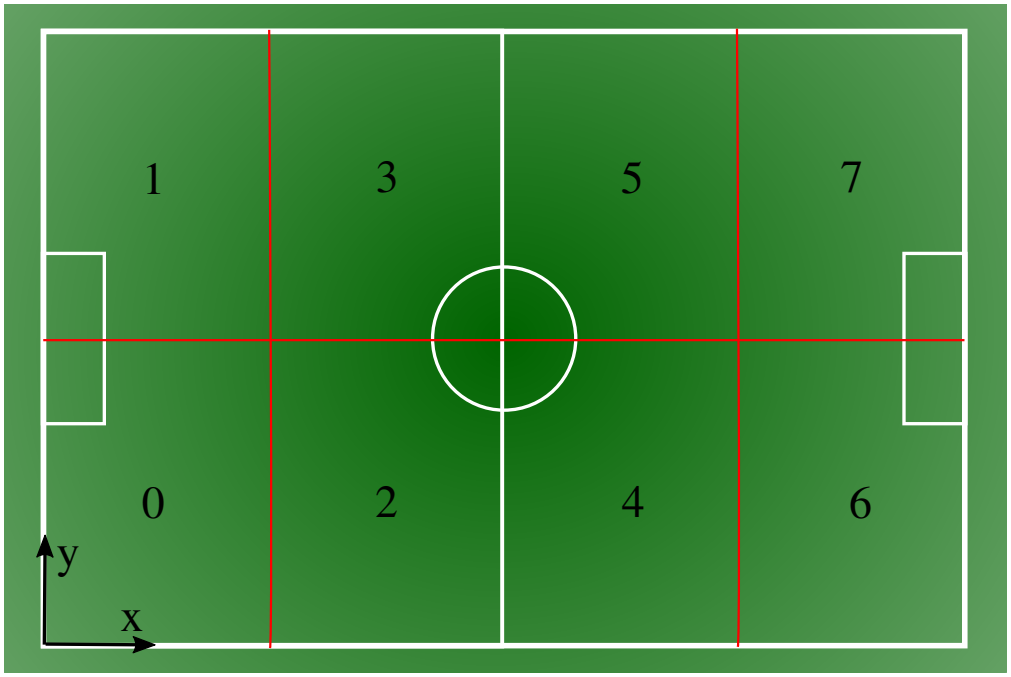


Fig. 4.2: Seções criadas afim de reduzir o número total de dígitos transmitidos.

4.3 HAAR CASCADE

Inicialmente, propôs-se a utilização de técnicas de reconhecimento de objetos de modo a se detectar o robô transmissor, para que, em seguida, fosse possível a segmentação de seus LEDs e a correta leitura da mensagem.

Esta seção apresentará as técnicas utilizadas para se detectar o robô em uma imagem. Primeiramente, irá se descrever os métodos e ferramentas utilizados para a aquisição de dados. A seguir se apresenta os passos necessários para o treinamento de um classificador em cascata utilizando os conceitos da Sec. 2.2.1.2.

4.3.1 Aquisição de Dados

Tanto ao se utilizar os métodos de classificação em cascata como na aplicação de qualquer outra técnica de visão computacional é importante o foco na aquisição de imagens e no seu pré-processamento. Com isso em mente, foram gerados alguns bancos de dados com imagens do robô, além de imagens negativas, requeridas para o treinamento. Todas as imagens positivas foram gravadas a partir das câmeras presentes na cabeça de um robô quando este observa o outro com resolução de 640x480 pixeis.

O primeiro banco gerado foi obtido através da gravação de vídeos com o robô em movimento em um campo de futebol. As imagens obtidas nesse momento foram utilizadas para testes preliminares

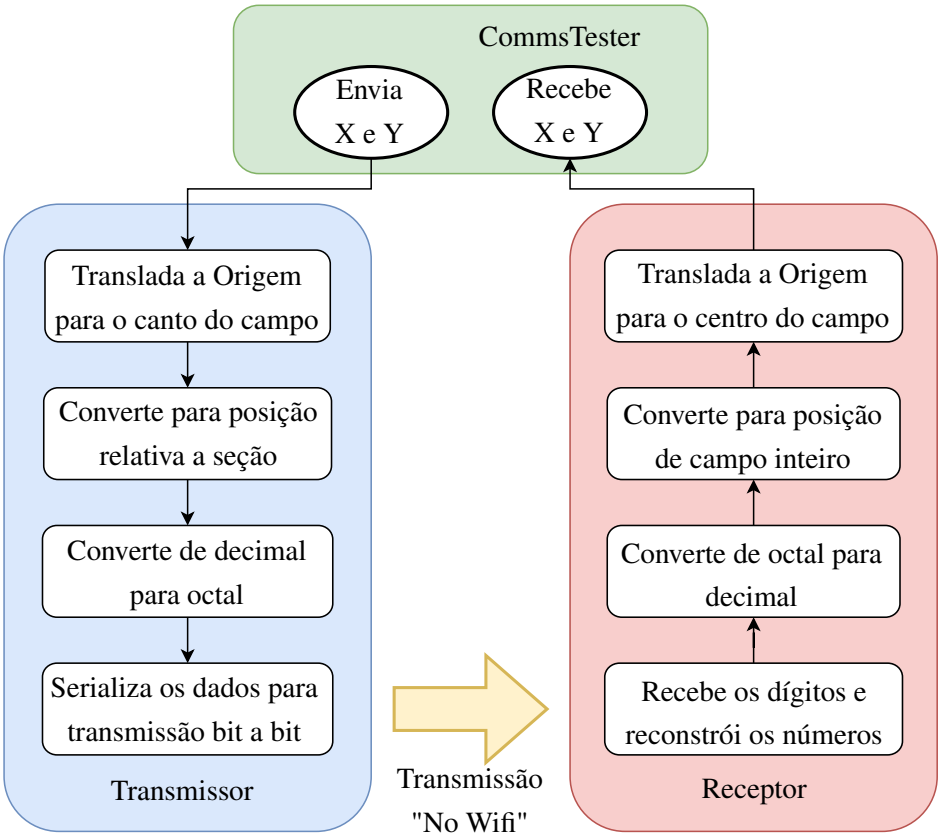
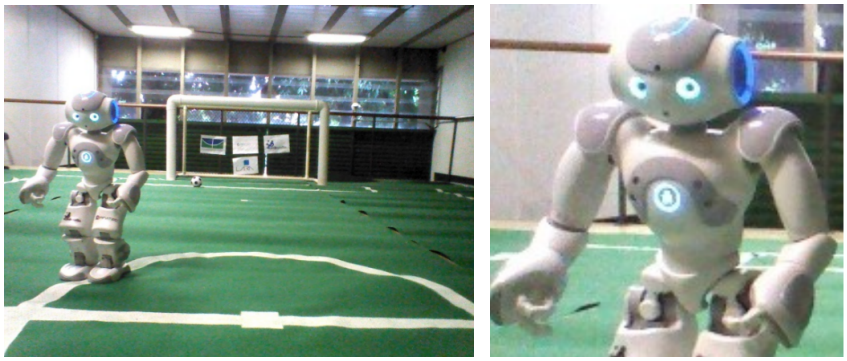


Fig. 4.3: Diagrama mostrando as etapas de codificação e decodificação da mensagem.

na detecção de corpo inteiro de um robô e foram validadas também com o robô em movimento no campo. As imagens para o treinamento foram obtidas através da amostragem do vídeo e seu pré-processamento se deu por meio de cortes mantendo sempre a mesma proporção entre a altura e a largura do corte. A Fig. 4.4 mostra um exemplo da imagem não processada e do corte subsequente.



(a) Imagem Obtida do Robô, não pré processada. (b) Imagem Cortada para aplicação no Haar Cascade.

Fig. 4.4: Exemplos de Imagens do banco de dados com o robô inteiro em movimento

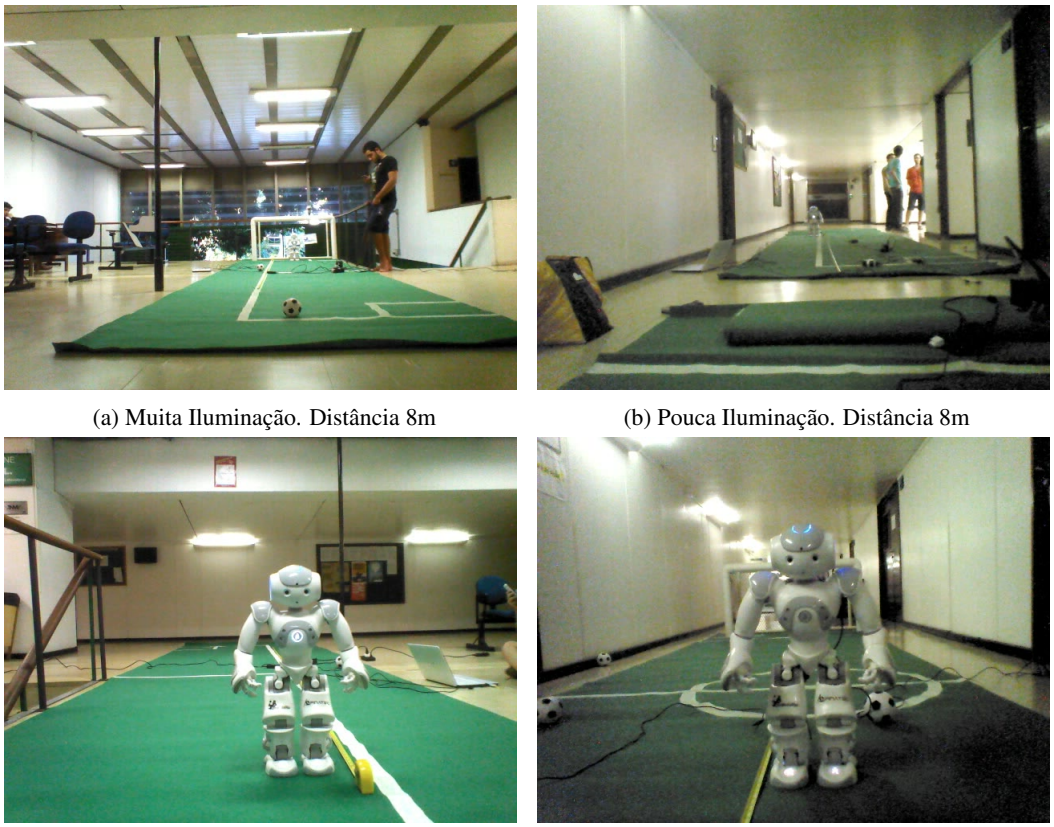
Visando obter resultados melhores, mais vídeos com o robô em movimento pelo campo foram gravados e amostras de imagens foram obtidas. A partir das amostras foram feitos cortes separando algumas partes do robô, mais especificamente, criou-se um banco de dados para as pernas, braços,

cabeça e peito do robô. Cada conjunto foi submetido a um treinamento diferente. A Fig. 4.5 mostra exemplos de amostras utilizadas no treinamento.

(a) Cabeça (b) Peito (c) Pernas (d) Braço

Fig. 4.5: Exemplos de amostras positivas utilizadas em cada treinamento.

Finalmente, visando o experimento proposto, de modo a refinar o treinamento para a utilização no desafio proposto, foram obtidas novas amostras com o robô parado em diversas posições pré-definidas do campo de futebol. Para esse banco de dados, as imagens foram classificadas pela distância e pelo diferente nível de iluminação, variações específicas entre amostras em uma mesma distância foram obtidas através dos LEDs que foram colocados para piscar em cores diferentes. Estas imagens foram então processadas e cortadas as regiões de interesse da cabeça e do peito.



(a) Muita Iluminação. Distância 8m

(b) Pouca Iluminação. Distância 8m

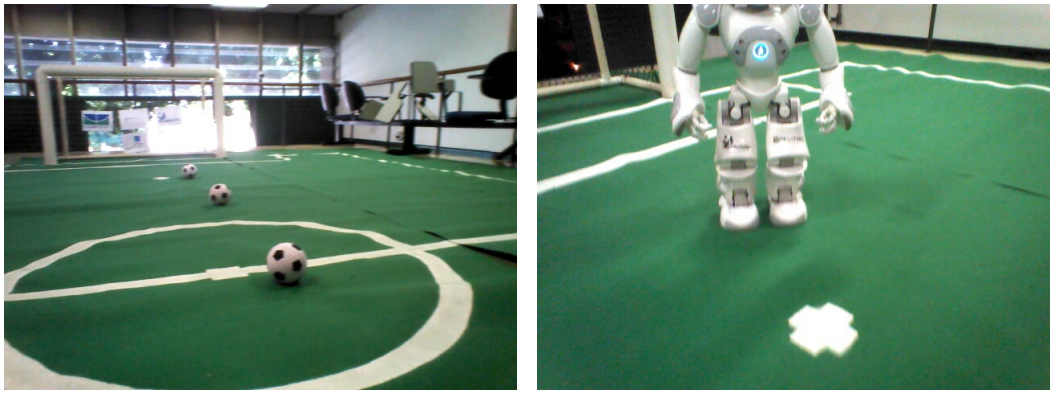
(c) Muita Iluminação. Distância 1m

(d) Pouca Iluminação. Distância 1m

Fig. 4.6: Imagens utilizadas no treinamento, variando-se distância e iluminação.

Para o banco de dados de imagens negativas foram utilizadas tanto fotos do campo de futebol com

diversos objetos que não fossem o NAO, como imagens não relacionadas obtidas na internet. Além disso, utilizou-se fotos de robôs que não possuíam a parte de um dado treinamento, de forma a evitar falsos positivos em diferentes partes do robô. A Fig. 4.7 mostra exemplos de amostras negativas.



(a) Imagem negativa utilizada em todos os treinamentos.

(b) Imagem negativa utilizada no treinamento de detecção de cabeças.

Fig. 4.7: Exemplos de imagens negativas utilizadas nos treinamentos.

4.3.1.1 Video Recorder

Deve-se notar que todas as imagens utilizadas no treinamento e nos testes subsequentes foram obtidas utilizando a ferramenta Video Recorder, de forma a garantir que o resultado fosse otimizado para imagens obtidas pelo robô.

Desse modo, para auxiliar na captura das imagens foi criado uma interface para a aquisição de gravações feitas utilizando-se qualquer uma das câmeras do robô diretamente. Esta ferramenta foi desenvolvida utilizando-se o framework QT [40] , que gerencia tanto a interface gráfica, como o processamento em paralelo, através da classe QThreads. A Fig. 4.8 mostra a interface do programa criado.

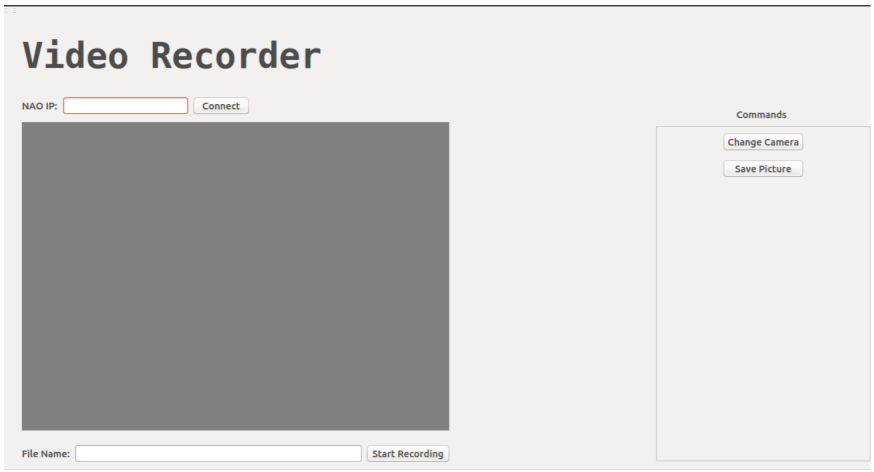


Fig. 4.8: Interface do Video Recorder

Essa interface conta com campos para digitar o nome do arquivo a ser salvo, escolha entre uma das

duas câmeras do NAO e um campo para inserir o IP do robô que está sendo utilizado (para a captura de imagens e acesso a memória do robô estão sendo utilizadas funções da NAOqi, disponibilizadas pelo desenvolvedor).

4.3.2 Treinamento

Com as imagens já obtidas e devidamente cortadas foram rodadas as funções para o treinamento do Haar Cascade presentes na distribuição 2.4.9 do OpenCV.

Para sumarizar o processo de treinamento, tem-se os seguintes passos:

1. Obtenção das imagens positivas.
2. Cortes em cada imagem de forma a selecionar o objeto a ser detectado. Deve-se notar que os cortes devem manter a mesma proporção de altura por largura em todas as amostras.
3. Criação de um arquivo .txt com o caminho para cada imagem positiva e as coordenadas em que o corte foi feito.
4. Obtenção das imagens negativas.
5. Criação de um .txt com o caminho para cada imagem negativa.
6. Criação de um arquivo .vec com a função opencv_createsamples. Esse arquivo criará um vetor com as imagens positivas a serem utilizadas.
7. Utilização da função opencv_traincascade para realizar o treinamento. Esta função aceita diversos parâmetros, como taxa de acerto mínima, quanta memória se deseja utilizar, máxima porcentagem de falsos positivos, tipo de classificador, etc [17].

4.4 SEGMENTAÇÃO DE LEDS NA IMAGEM

A forma de comunicação escolhida se baseia em utilizar diferentes cores nos LEDs, localizados na cabeça e no peito do robô, para transmitir dígitos na base octal conforme Sec. 4.2. Para que a comunicação tarefa seja executada, o robô receptor deve ser inicialmente capaz detectar o robô transmissor. Dado que os resultados do treinamento com o Haar Cascade não se mostraram suficientes para o desafio proposto (descrito na Sec. 1.1.1.1), visto que o robô só era identificado a distâncias inferiores a 2m, fez-se necessário a implementação de uma técnica diferente, de forma a segmentar cada LED antes da transmissão. Portanto, criou-se uma fase de calibração, a qual será descrita a seguir.

A estratégia adotada para a segmentação consiste em se alterar as cores dos LEDs do transmissor em uma frequência de aproximadamente 7Hz, enquanto o receptor obtém imagens da câmera a uma frequência de 4Hz. Considerando que os LEDs possuem 8 estados diferentes, isto é, 7 cores mais o estado desligado, pode-se garantir que as cores dos LEDs serão diferentes, dadas duas imagens obtidas

em sequência pelo receptor. Desta forma pode-se utilizar um algoritmo de detecção de diferenças entre as imagens.

O algoritmo de segmentação de LEDs na imagem descrito a seguir é demonstrado na Fig. 4.9 e no Algoritmo 4.1. Inicialmente, o receptor obtém duas imagens e executa a subtração entre elas. A imagem obtida é então convertida para a escala de cinza. O resultado desta operação é uma imagem a qual terá valores próximos de 0 nos pixeis que possuem valores iguais nas duas imagens originais. Logo, utiliza-se um filtro binário, onde valores entre 0 e 10 sejam considerados como 1 e valores entre 11 e 255 sejam considerados como 0. Assim, obtêm-se uma imagem binária, a qual contém todos os pixeis que se mantiveram iguais nas duas imagens originais. Esta imagem é armazenada e as operações descritas acima são executadas novamente, utilizando-se duas . Uma nova imagem é então obtida e executa-se uma operação OR bit-a-bit com a imagem binária anterior. Logo, pixeis que possuam valor 1 em pelo menos uma das duas imagens binárias recebem também o valor 1, enquanto pixeis que possuam valor 0 em ambas as imagens permanecem como 0. Desta forma, a imagem binária resultante mostra todos os pixeis que se alteraram em pelo menos uma das execuções do algoritmo descrito.

Algoritmo 4.1 Segmentação de LEDs na imagem.

```
diffTotal ← null;
for i ← 0 to 5 do
   Obtém Imagem1;
   Obtém Imagem2;
   diff = Imagem1 - Imagem2;
   Converte diff para escala de cinza;
   Converte diff para imagem binária;
   diffTotal ← OR-bit-a-bit(diffTotal, diff);
end for
Inverte diffTotal;
```

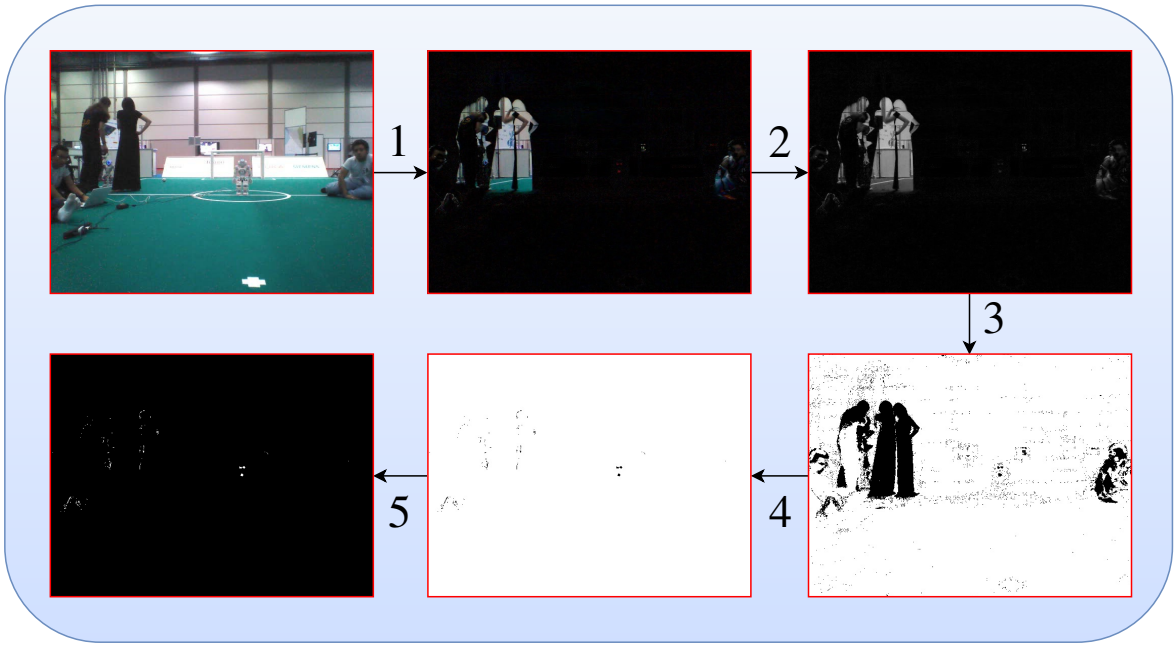


Fig. 4.9: Algoritmo utilizado para segmentação de LEDs na imagem. As operações estão numeradas da seguinte forma. 1-Subtração das imagens. 2-Conversão para escala de cinza. 3-Filtro binário. 4-OR bit-a-bit. 5-Inversão da máscara.

Após uma quantidade definida de iterações, a imagem binária resultante é invertida, de modo a obter uma imagem que filtra apenas os pixeis que sofreram alguma alteração em todas as imagens.

A Fig. 4.10 mostra o resultado após cinco iterações. Percebeu-se que, devido aos ruídos das imagens e ao tamanho dos LEDs, após um número superior a seis iterações, os pixeis referentes aos LEDs também eram removidos. Portanto, optou-se por utilizar sempre cinco iterações. Nota-se também que ainda existem ruídos indesejados na imagem resultante, os quais, em sua maioria, foram removidos utilizando-se operações morfológicas. Pode-se também utilizar técnicas de reconhecimento de objetos, de forma a limitar a área na qual o algoritmo de segmentação de LEDs na imagem atua, reduzindo o tamanho do *background* e, consequentemente, a quantidade de ruídos.

(a) Exemplo de imagem obtida pelo robô.

(b) Imagem binária após 1 iteração.

(c) Imagem binária após 2 iterações.

(d) Imagem binária após 3 iterações.

(e) Imagem binária após 4 iterações.

(f) Imagem binária após 5 iterações.

Fig. 4.10: A figura 4.10a exemplifica a visão do robô em um dado momento da transmissão. As outras figuras demonstram 5 iterações da segmentação de LEDs na imagem. As imagens binárias foram invertidas para melhor visualização.

4.5 TRANSMISSÃO DE DADOS

Esta seção apresentará a descrição do algoritmo utilizado na aplicação descrita, isto é, a comunicação sem estrutura de redes.

4.5.1 Calibração

A calibração se trata do algoritmo realizado antes de qualquer transmissão no desafio. Ela é utilizada para se segmentar os LEDs que serão usados na etapa seguinte. O algoritmo se inicia após os robôs serem corretamente posicionados como descrito na Fig. 1.3. Deve-se então pressionar o botão no peito de ambos os NAOs para se dar início à fase de calibração.

Inicialmente, deve-se criar uma área de interesse ao redor do robô transmissor, de forma a diminuir os ruídos gerados pelo *background*. Devido aos resultados não satisfatórios do treinamento utilizando Haar Cascade, optou-se por utilizar a técnica de Match Template, descrita na Sec. 2.2.1.1, utilizando uma foto das pernas do robô tirada no local de competição. Desta forma, após a identificação do robô, cria-se uma região de interesse, a qual será utilizada nos algoritmos subsequentes.

Executa-se então o algoritmo de segmentação de LEDs descrito na Sec. 4.4. Com a imagem resultante, após a execução de operações morfológicas de forma a eliminar os ruídos existentes, utiliza-se um algoritmo de detecção de blobs[1] o qual retorna as coordenadas dos três LEDs. Deve-se então executar uma rotina de categorização de acordo com a posição, dado que existem três LEDs e as suas posições relativas são conhecidas. As coordenadas são então salvas na seguinte ordem: peito, olho direito e olho esquerdo. Tais coordenadas serão utilizadas na leitura da mensagem.

O fim da etapa de calibração acontece quando o transmissor altera a cor de seus três LEDs para vermelho. A etapa de transmissão se inicia assim que o CommsTester enviar as posições para o transmissor.

4.5.2 Transmissão da Mensagem

Para a correta transmissão de dados, fez-se necessário a criação de um protocolo de cores. A escolha das três cores necessárias foi baseada nos valores de Hue[2] das 8 opções existentes. Dados os valores mostrados na Tab. 4.2 as cores escolhidas foram vermelho para representar a transição entre dois dígitos, ciano para representar o valor 1 e amarelo para representar o valor 0.

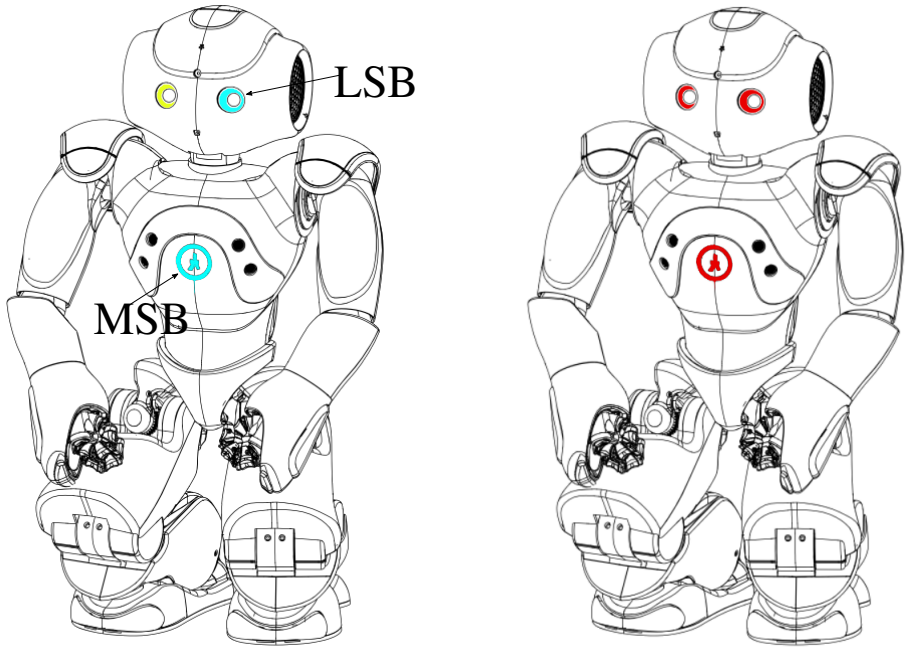
Tab. 4.2: Valores mínimos e máximos de Hue para cada cor de LED detectada pelo robo.

Cor	Hue Mínimo	Hue Máximo
Amarelo	35	45
Azul	125	135
Branco	100	120
Ciano	95	105
Verde	70	90
Vermelho	170	190
Violeta	130	140

[1]Detecção de regiões em imagens a partir de diferenças em propriedades como luminosidade e cor comparadas com regiões adjacentes.

[2]Propriedade de matiz de uma imagem

O transmissor, ao receber os dados do CommsTester, executa as etapas de codificação mostradas na Fig.4.3. Cada dígito octal é então convertido para números binários, no qual o LED do peito irá transmitir o bit mais significativo, enquanto o do olho esquerdo será responsável pelo bit menos significativo. Os LEDs assumem então a cor ciano ou amarelo de acordo com o valor do bit o qual são responsáveis por transmitir. Após 0.75 segundos, todos os LEDs assumem a cor vermelha por 0.25 segundos, para informar ao receptor que um novo dígito será enviado. Estes passos são repetidos até que os 9 dígitos sejam enviados. A Fig.4.11 contém exemplos de cores dos LEDs nas fases de transmissão de dígitos e de transição.



(a) LEDs representando o valor 5_8, isto é, 101_2.

(b) LEDs representando a transição entre a transmissão de dois dígitos diferentes.

Fig. 4.11: Diferentes configurações assumidas pelos LEDs do robô ao longo da etapa de transmissão. (*adaptado de: http://doc.aldebaran.com/*)

O receptor obtém imagens a 20 *frames* por segundo. Em cada imagem obtida, monitora-se os valores de Hue médios nas coordenadas de cada LED, obtidas na etapa de calibração. Estes valores médios são salvos em um vetor enquanto forem diferentes de vermelho. Assim que o estado de transição é detectado, isto é, detecta-se a cor vermelha nos três LEDs do transmissor, o receptor percorre o vetor com os valores salvos e a cor com o maior número de ocorrências é escolhida como valor mais provável. Os valores de maior probabilidade em cada LED são armazenados e o receptor aguarda o fim do período de transição. Os passos descritos acima são repetidos até o receptor possuir 9 dígitos salvos. Neste instante, ele executa o algoritmo de decodificação descrito na Fig. 4.3, envia as posições resultantes ao CommsTester e então aponta para a posição aproximada no campo.

5 RESULTADOS: COMUNICAÇÃO VISUAL

"Once You replace
negative toughts with positive ones, you will start having positive results"

Willie Nelsón

5.1 INTRODUÇÃO

Nesta seção será feita uma análise do sistema de comunicação visual descrito no Cap. 4. Aqui serão apresentados os resultados obtidos tanto na detecção de objetos quanto na transmissão de mensagem.

5.2 DETECÇÃO DE OBJETOS

Nesta seção será feita uma análise qualitativa do método de detecção de objetos que baseado no Haar Cascade. Primeiramente será feita uma análise do treinamento realizado com o corpo inteiro do robô, em seguida será feita a análise separada tanto da detecção do braço quanto da cabeça e do peito.

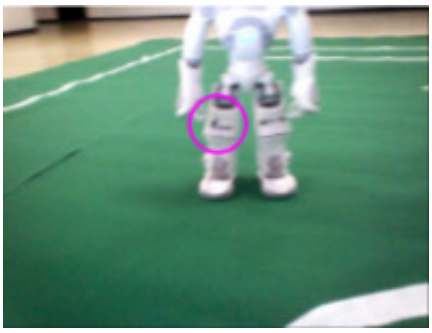
De fato, é interessante ressaltar que a análise dos diversos treinamentos feitos além de literatura relacionada, mostra que esse tipo de método de classificação pode ser utilizado para a detecção de objetos não rotacionados ou simétricos, sendo em alguns casos aplicável para a detecção de um robô. Entretanto, a medida que cada teste foi feito, a busca por uma solução robusta para detecção de robôs no desafio proposto pareceu não ser possível através do Haar Cascade. Desse modo optou-se por utilizar métodos como o template matching e a segmentação de LEDs para garantir a confiança no sistema.

5.2.0.1 Corpo Inteiro do Robô

Para o treinamento da detecção de corpo inteiro do robô foi utilizado um banco de dados com 105 imagens, todas dimensionadas de forma quadrada. Os valores de entrada do treinamento para taxa de acerto mínimo foi de 99.5% e o máximo de falsos positivos foi de 50%. Esses valores são utilizados pelo treinamento de modo a definir quais classificadores utilizar ou descartar. A Fig.5.1 mostra alguns dos resultados obtidos.

(a) Robô em movimento

(b) Detecção do robô parado



(c) Detecção de falso positivo

Fig. 5.1: Resultados da detecção do robô inteiro.

Ao se validar o treinamento com vídeos gerados, obteve-se como resultado que o robô quase nunca era detectado em movimento, como visto na Fig. 5.1a. Dependendo da distância, o NAO pode ser detectado quando estava parado, como mostra a Fig. 5.1b, e além disso, houve a ocorrência de falsos positivos principalmente nas pernas do robô, quando este se encontrava mais próximo como mostra a Fig. 5.1c.

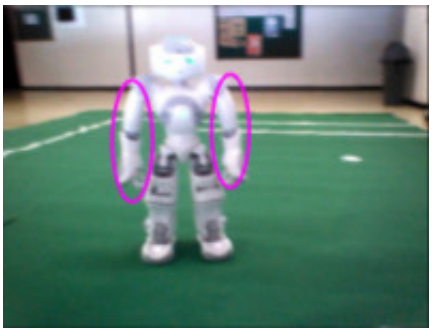
Como o banco de dados utilizado era pequeno, a detecção só ocorria quando o robô parava em certas poses.

5.2.0.2 Braço

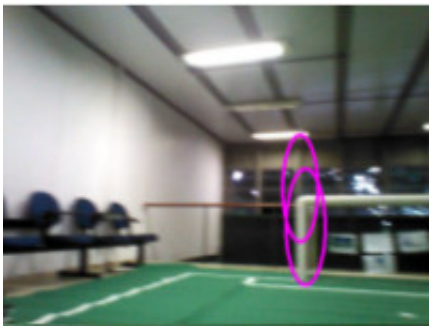
Para o braço também foi utilizado o Haar Cascade com a taxa de acerto mínimo de 99.5% e o máximo de falsos positivos de 50%. Neste teste em cada imagem os braços foram cortados separadamente, mas visando realizar a detecção do que seria a imagem refletida o treinamento foi feito com ambos ao mesmo tempo. A Fig. 5.2 mostra alguns dos resultados obtidos.

Neste caso nota-se mais uma vez que no robô parado de forma frontal é possível detectar de forma satisfatória ambos os braços, como pode-se ver na Fig. 5.2a. Já nas outras duas imagens nota-se uma quantidade maior de falsos positivos, dessa vez detectados na trave do gol e no próprio robô.

Dado que o banco de dados desta vez possuía 475 imagens, o que era um número consideravelmente maior que o inicial, não se pode atribuir o problema a invariância das imagens. De fato, como se vê na literatura, os classificadores treinados com Haar Cascade tendem a trabalhar muito bem para diferentes condições de iluminação e distância, mas os resultados para variações na posição do objeto, como rotações, tendem a gerar maus classificadores [41]. Algumas das propostas feitas para



(a) Ambos os braços detectados



(b) Falso positivo encontrado nas traves do Gol

(c) Falso positivo no robô inteiro

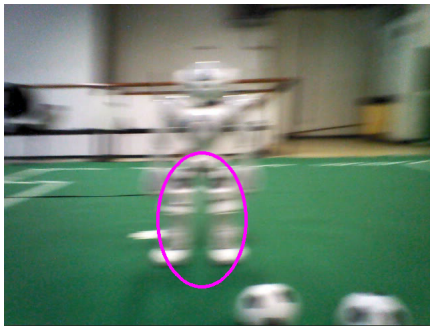
Fig. 5.2: Detecção de braços do robô.

solucionar este problema são realizar treinamentos em diferentes intervalos de ângulo. No entanto, esta solução não é necessária para o experimento proposto no desafio, uma vez que um robô sempre estará posicionado de frente para o outro. Ademais, variações na rotação não são a causa dos falsos positivos, os quais também deseja-se eliminar.

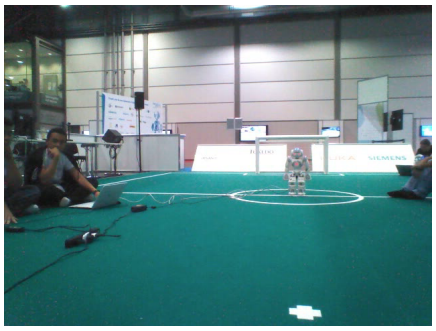
5.2.0.3 Pernas

O treinamento para a detecção de pernas, visto na Fig. 5.3, foi feito com taxa de acerto mínimo foi de 99.5% e o máximo de falsos positivos foi de 50% em um banco de 377 imagens. Estes resultados se mostram os melhores resultados dentre os treinamentos já explicitados.

(a) Perna encontrada



(b) Perna encontrada durante movimento



(c) Perna não encontrada

Fig. 5.3: Detecção da perna do robô.

Nota-se, primeiramente, a Fig. 5.3a e a Fig. 5.3b, aqui tanto com movimento quanto com a imagem estática se encontrou a perna de maneira satisfatória sem a geração de falsos positivos, além disso havia a perda do objeto somente em alguns quadros dos vídeos, mostrando constância nos resultados. De fato, as imagens de perna se mostram eficientes para a detecção de robô visto que elas tem padrões bem definidos além do fundo, em um campo de futebol, constante (sempre verde, sem muito ruído).

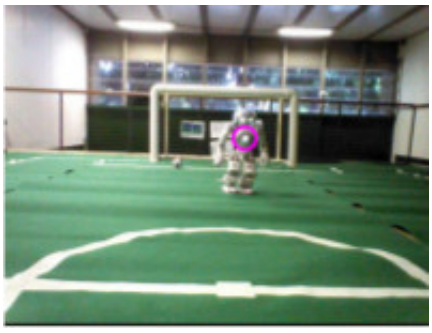
Finalmente, como visto na Fig. 5.3c, não há detecção quando o robô está a distâncias de mais de 1.5m, o que mostra o maior ponto negativo desta detecção.

5.2.0.4 Placa frontal

Foram feitos testes para a detecção da placa no peito do humanoide e para a cabeça.

Para este treinamento, foi utilizado um banco de dados com 4700 imagens, e como uma alternativa para o Haar, foi utilizado algoritmo LBP[1], com taxa de acerto mínimo de 90% e o máximo de falsos positivos de 30%. Os resultados obtidos são mostrados na Fig. 5.4

[1]Local Binary Patterns, treinamento similar ao Haar Cascade, entretanto mais rápido e com resultados inferiores.



(a) Robô detectando em movimento

(b) Robô detectado na posição frontal

Fig. 5.4: Detecção da placa no peito do robô.

Neste caso é visto que o peito do robô foi detectado com maior eficiência, gerando poucos falsos positivos, como esperado com a mudança dos parâmetros e do treinamento mais intenso. Entretanto o robô continuou não sendo encontrado a distâncias maiores que 5m, o que foi validado no próprio material usado para o treinamento.

5.2.0.5 Cabeça

Foi feito também um treinamento para a detecção da cabeça do robô, visto na Fig. 5.5. Os parâmetros utilizados foram de acerto mínimo de 90% e o máximo de falsos positivos de 30% , entretanto, nota-se que os resultados obtidos foram insatisfatórios.

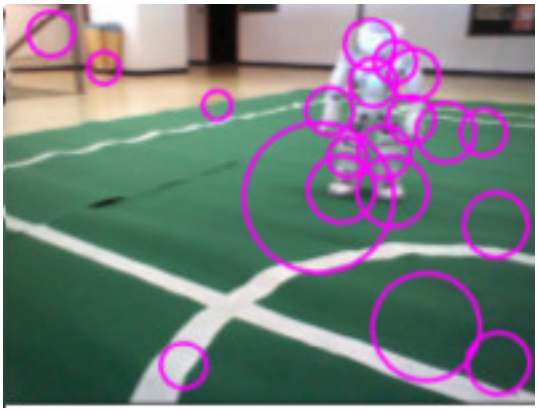


Fig. 5.5: Detecção da cabeça do robô.

Neste caso notou-se que diversas imagens estavam muito pequenas ou borradas, e portanto foram ignoradas pelo classificador, fazendo assim um treinamento pouco confiável, que gerava muitos falsos positivos.

5.3 TRANSMISSÃO DE MENSAGENS

Para validar o método de transmissão de dados descrito na Sec. 4.5 dois robôs, um transmissor e um receptor, foram posicionados no campo de futebol como na Fig.4.1. Com o software CommsTester, posições aleatórias foram geradas, o robô transmissor as traduziu para o padrão definido e enviou a partir de seus LEDs. Como programado, o robô receptor recebeu a mensagem e apontou para a posição do campo referente ao valor obtido. A Tab. 5.1 mostra os dados enviados e recebidos. Note que, não foram utilizadas técnicas de detecção de objetos, apenas a segmentação de LEDs descrita na Sec. 4.4.

Tab. 5.1: Tabela de dados enviados pelo transmissor (a esquerda) e recebidos pelo receptor (a direita)

Valor Enviado					Valor Recebido				
X	Y	Sect.	X Oct	Y Oct	X	Y	Sect.	X Oct	Y Oct
-429	2896	3	3435	5520	-429	2896	3	3435	5520
-2788	-2788	0	3260	0324	-2788	-1300	0	3260	3244
1996	819	3	0376	1463	3843	2463	7	3071	4637
2823	-2796	6	1075	0314	2738	1636	7	0750	3144
233	1921	5	0351	3601	233	1921	5	0351	3601
-1517	-2491	2	1335	0775	-1517	-2491	2	1335	0775
-358	1241	3	3544	2331	603	1252	3	5445	2344
1712	9	5	3260	0011	5024	1	7	5326	0001
-2690	-1296	0	3422	3250	-2690	-1296	0	3422	3250
2820	-1560	6	1072	2640	2820	-1560	6	1072	2640
-1683	-1370	2	1067	3136	3368	3787	7	2136	7313
-2090	18	3	0240	0022	-2074	0151	3	0260	0227
-1877	393	3	0565	0611	-1877	393	3	0565	0611

Analisando a Tab. 5.1, nota-se que foram feitos treze envios diferentes e desse modo foram enviados 117 valores octais ao todo. Visando se analisar melhor a taxa de transmissão foi calculada as porcentagens de acerto como na Tab. 5.2.

Tab. 5.2: Tabela de porcentagem de acerto da tarefa

	%
Acerto por dígito	64.10
Acerto por dígito com offset	88.89
Acerto da mensagem	46.15

A detecção dos dígitos era computada quando o robô acertava os três LEDs no momento correto, isto ocorreu com uma taxa de acerto de 64.10% dos 117 dígitos enviados. Muitas vezes, entretanto, um dos dígitos era perdido, e, mesmo que houvessem acertos subsequentes estes ficariam fora de ordem. Do mesmo modo, caso fosse detectado um dígito quando ainda na fase de calibração, todos

os acertos subsequentes também ficariam fora de ordem. Assim introduziu-se um *offset*, de modo que as detecções corretas ficassem na posição correta.

A taxa de acerto com o *offset* foi de 88.89%. Mesmo com os dígitos deslocados, a mensagem foi detectada de maneira errada, desse modo, a taxa de acerto total foi de 46.15%.

6 DESENVOLVIMENTO: CONTROLE COOPERATIVO

"Any sufficiently
advanced technology is indistinguishable from magic."

Arthur C. Clarke

6.1 INTRODUÇÃO

Visando criar ferramentas de cooperação voltadas a robótica, foram criados diversos controladores que permitiam a realização de tarefas que envolvam múltiplos manipuladores. Assim, nesta seção será descrita a estrutura computacional e matemática usada no desenvolvimento de cada controlador, além de explicar a configuração das plataformas utilizadas para a validação de tais controladores.

6.2 CINEMÁTICA DO NAO

O problema de cinemática do NAO envolve a descrição de sua estrutura mecânica por meio de uma representação tridimensional, no caso os parâmetros de Denavit-Hartenberg.

6.2.1 Parâmetros de Denavit-Hartenberg dos Braços

Antes de se iniciar a definição dos parâmetros DH, é importante notar que eles devem ser feitos considerando a posição zero do robô, isto é, a posição assumida quando todas as suas juntas são ajustadas para zero. Além disso, deve-se escolher um sistema de coordenadas de referência que será utilizado como base do robô. A Fig. 6.1 mostra o robô na posição zero além de definir a base.

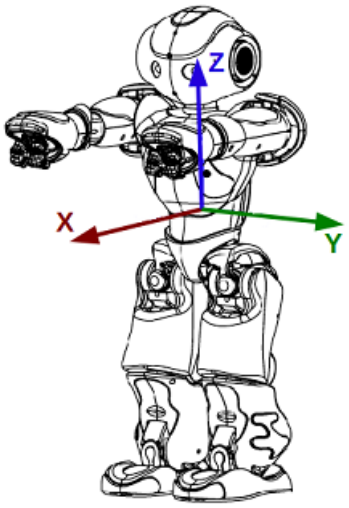


Fig. 6.1: Sistema de coordenadas base e posição zero.

Embora a fabricante do robô forneça os parâmetros DH para todas as cadeias cinemáticas, Kofinas [35] afirma que tais valores estão incorretos, logo, optou-se por calcular os parâmetros manualmente. De fato ao se calcular tais parâmetros notou-se que eles eram distintos dos fornecidos.

O braço esquerdo do robô é composto por cinco juntas, logo, deve-se obter cinco conjuntos de parâmetros DH. Partindo da base, deve-se transladar o sistema ao longo do eixo z e então rotacioná-lo $-\frac{\pi}{2}$ em torno do eixo x. Nota-se que, apesar do sistema ainda estar em uma posição diferente da primeira junta, o eixo z já se encontra alinhado com o eixo de rotação da junta. Para a segunda junta deve-se primeiro transladar ao longo do eixo z do sistema de coordenadas anterior e então executar uma rotação de $\frac{\pi}{2}$ em torno do eixo x. Para a articulação subsequente, deve-se alinhar o eixo z com o eixo de rotação da junta de guinada do cotovelo. Portanto, deve-se inicialmente rotacionar o sistema $\frac{\pi}{2}$ em torno do eixo z, transladar ao longo do eixo x e rotacionar $\frac{\pi}{2}$ em torno do eixo x. Para a próxima junta, se faz necessário mover o sistema para o cotovelo, logo, deve-se primeiramente transladar ao longo do eixo z e então rotacionar $-\frac{\pi}{2}$ em torno do eixo x, de forma a alinhar o eixo z com o eixo de rotação da junta. Para a última junta, necessita-se apenas de uma rotação de $\frac{\pi}{2}$ em torno do eixo x. Por fim, deve-se executar as seguintes transformações para posicionar o sistema de coordenadas no efetuador final. Os passos necessários são: transladar ao longo do eixo x, rotacionar $-\frac{\pi}{2}$ em torno do eixo x e por fim rotacionar$-\frac{\pi}{2}$ em torno do eixo z. Desta forma, posicionamos o sistema no final do efetuador, mantendo a mesma orientação da base, considerando a posição zero. Os passos listados são resumidos na Tab. 6.1 na forma de parâmetros DH.

Tab. 6.1: Parâmetros DH do braço esquerdo. ShoulderOffsetZ e ShoulderOffsetY denotam o deslocamento necessário para mover-se o sistema de coordenadas da base para o ombro em z e y, respectivamente. ElbowOffsetY representa a diferença ao longo do eixo y da base entre as juntas do ombro e as juntas do cotovelo. UpperArmLength e LowerArmLength denotam o comprimento do braço e do antebraço do robô. Todas as constantes aqui utilizadas se encontram na Tab. 3.1.

Junta	θ	d	a	α
Base (dummy)	0	ShoulderOffsetZ	0	$-\frac{\pi}{2}$
ShoulderPitch	0	ShoulderOffsetY	0	$\frac{\pi}{2}$
ShoulderRoll	$\frac{\pi}{2}$	0	ElbowOffsetY	$\frac{\pi}{2}$
ElbowYaw	0	UpperArmLength	0	$-\frac{\pi}{2}$
ElbowRoll	0	0	0	$\frac{\pi}{2}$
WristYaw	0	LowerArmLength	0	$-\frac{\pi}{2}$
End Effector (dummy)	$-\frac{\pi}{2}$	0	0	0

Para o braço direito, a maioria dos passos são idênticos àqueles utilizados para determinar os parâmetros DH do braço esquerdo. As únicas diferenças são os sinais dos deslocamentos ao longo do eixo y do sistema de coordenadas da base, ou seja, deslocamento do ombro em y e deslocamento do cotovelo em y. Os parâmetros DH do braço direito são mostrados na Tab. 6.2.

Tab. 6.2: Parâmetros DH do braço direito.

Junta	θ	d	a	α
Base (dummy)	0	ShoulderOffsetZ	0	$-\frac{\pi}{2}$
ShoulderPitch	0	-ShoulderOffsetY	0	$\frac{\pi}{2}$
ShoulderRoll	$\frac{\pi}{2}$	0	-ElbowOffsetY	$\frac{\pi}{2}$
ElbowYaw	0	UpperArmLength	0	$-\frac{\pi}{2}$
ElbowRoll	0	0	0	$\frac{\pi}{2}$
WristYaw	0	LowerArmLength	0	$-\frac{\pi}{2}$
End Effector (dummy)	$-\frac{\pi}{2}$	0	0	0

6.3 SETUP EXPERIMENTAL

Esta seção irá descrever as ferramentas utilizadas, primeiramente em ambiente de simulação e em sequência na plataforma física, para o teste e implementação dos controladores propostos, bem como cada configuração e os algoritmos utilizados como base.

6.3.1 Simulação

De forma a validar os controladores implementados antes de testá-los no robô, configurou-se um ambiente de simulação utilizando a plataforma V-Rep, desenvolvido pela Coppelia Robotics.

Utilizou-se a implementação da biblioteca DQ_Robotics em conjunto com a API remota da plataforma V-Rep, ambas com suporte para Matlab, de forma que todo o controle fosse implementado diretamente no Matlab, enquanto o V-Rep era responsável apenas por fornecer o ângulo atual juntas e aplicar a saída do controlador.

Construiu-se a cena mostrada na Fig. 6.2, composta por um copo, o qual é utilizado como referência do controlador, um NAO que é o robô a ser controlado e uma base que serve apenas como suporte para o copo. Nota-se que, devido a existência de discrepâncias entre a localização das juntas do robô simulado e do robô real, foi necessário alterar os parâmetros DH da simulação, de forma a garantir o correto funcionamento do controlador. Os parâmetros alterados se encontram na Tab. 6.3 e na Tab. 6.4.

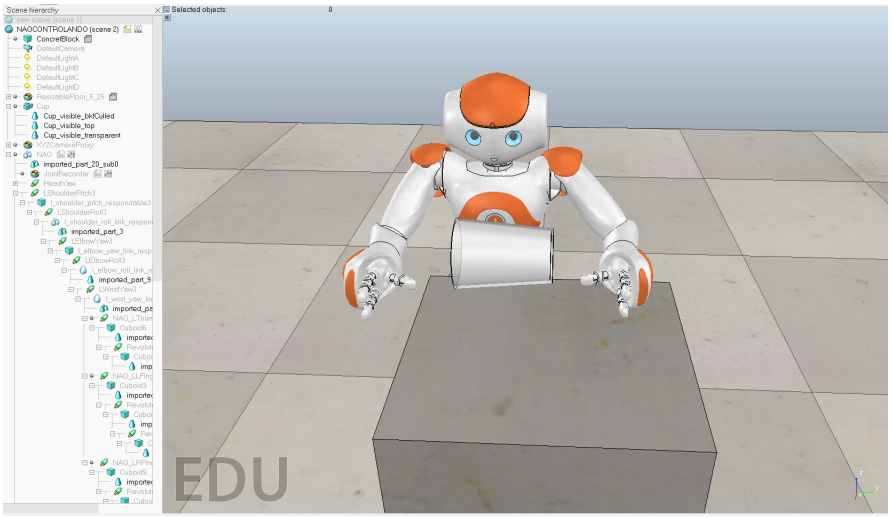


Fig. 6.2: Cena utilizada no ambiente de simulação.

Tab. 6.3: Parâmetros DH do braço esquerdo alterados para a simulação no V-REP devido as discrepâncias do robô real.

Junta	θ	d	a	α
Base (dummy)	0	ShoulderOffsetZ - 0.0186	-0.0026	$-\frac{\pi}{2}$
ShoulderPitch	0	ShoulderOffsetY	0	$\frac{\pi}{2}$
ShoulderRoll	$\frac{\pi}{2}$	0	ElbowOffsetY	$\frac{\pi}{2}$
ElbowYaw	0	UpperArmLength	0	$-\frac{\pi}{2}$
ElbowRoll	0	0	0	$\frac{\pi}{2}$
WristYaw	0	LowerArmLength	0	$-\frac{\pi}{2}$
End Effector (dummy)	$-\frac{\pi}{2}$	0	0	0

Tab. 6.4: Parâmetros DH do braço direito alterados para a simulação.

Junta	θ	d	a	α
Base (dummy)	0	ShoulderOffsetZ - 0.0186	-0.0026	$-\frac{\pi}{2}$
ShoulderPitch	0	-ShoulderOffsetY	0	$\frac{\pi}{2}$
ShoulderRoll	$\frac{\pi}{2}$	0	-ElbowOffsetY	$\frac{\pi}{2}$
ElbowYaw	0	UpperArmLength	0	$-\frac{\pi}{2}$
ElbowRoll	0	0	0	$\frac{\pi}{2}$
WristYaw	0	LowerArmLength	0	$-\frac{\pi}{2}$
End Effector (dummy)	$-\frac{\pi}{2}$	0	0	0

O algoritmo base utilizado por todos os controladores testados no ambiente de simulação está descrito no Algoritmo 6.1

Algoritmo 6.1 Controle Cinemático em Simulação

Adiciona os *paths;*
Inicializa a comunicação com o V-Rep via API;
Obtém os *handles* das juntas do robô e do objeto de referência;
while erro > erro_min **do**
  Obtém valor atual das juntas do robô;
  Calcula a posição e orientação do efetuador;
  Calcula o erro e a Jacobiana;
  Obtém novos valores para as juntas e envia para o V-Rep;
end while

Em que paths é a localização da biblioteca, handles são as variáveis utilizadas para obter as informações advindas do simulador e erro_min é um valor pré definido para a condição de parada do controlador.

6.3.2 Plataforma Física

Para melhor validar os controladores projetados foi feita uma implementação na plataforma NAO, utilizando tanto o sistema ROS quando a biblioteca DQrobotics.

Aqui utilizou-se o sistema ROS, que como explicado na Sec. 3.5, é estruturado a partir de nós que se comunicam por serviços e mensagens. Desse modo o sistema foi estruturado a partir de quatro nós:

- um nó com o NAOqi_Driver, disponibilizado no repositório do ROS[1], para o acesso dos *encoders* das juntas de modo a ler e enviar valores. Este nó publica e lê do tópico *JointAngleVelocities* além de fazer a comunicação direta com os módulos do NAOqi no robô;
- um nó que lê os valores publicados pelo NAOqi_Driver e envia o obtido das juntas dos braços

[1]Pode ser acessado em http://ros-naoqi.github.io/naoqi_driver/

para o nó de controle;

- um nó que recebe os dados das juntas após a ação de controle e as publica no *JointAngleVelocities* para sua atuação. Nota-se que, por motivos de segurança, este bloco satura a velocidade em 50% de seu valor máximo;
- um nó de controle, que recebe as juntas e utilizando a biblioteca DQRobotics obtêm a sua representação em quatérnios duais. Esse nó chama, então, uma função de controle, que retorna um vetor de juntas atuado, esse vetor será então mandado para o nó de atuação.

A Fig. 6.3 mostra um fluxograma do sistema como um todo.

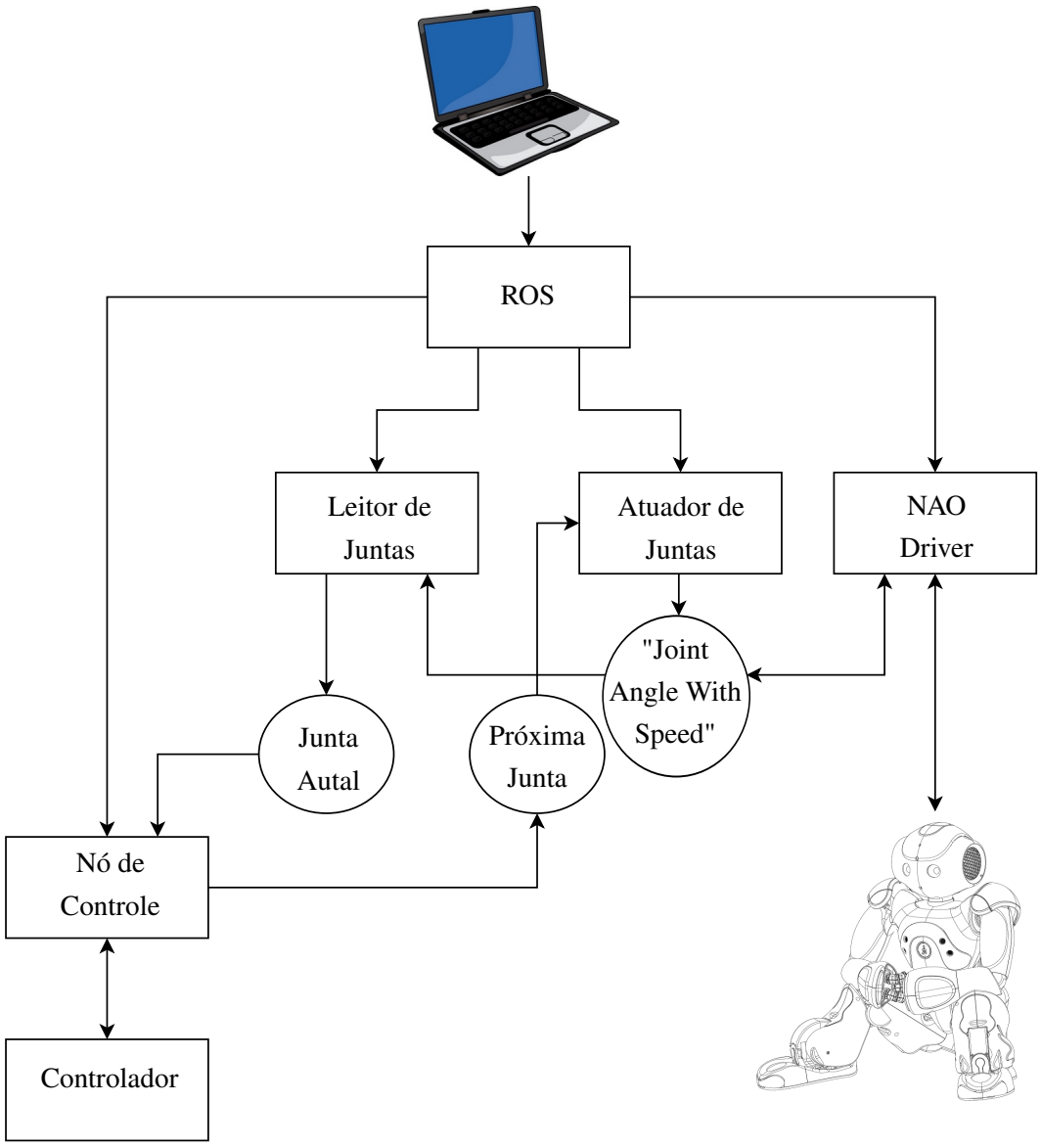


Fig. 6.3: Fluxograma do sistema implementado na plataforma física..

6.4 CONTROLADORES IMPLEMENTADOS

Dadas as técnicas de controle descritas na Sec. 2.6, a descrição de movimento de corpos rígidos, em particular com quatérnios duais unitários (Sec. 2.4) e a cinemática de manipuladores (Sec. 2.5) foram projetados diversos controladores cinemáticos no espaço de trabalho dos efetuadores, que permitiram a execução de diferentes tarefas. Nota-se que o projeto dos controladores feitos englobam controladores mais simples (com menos graus de liberdade) projetados previamente, como ilustrado na Fig. 6.4.

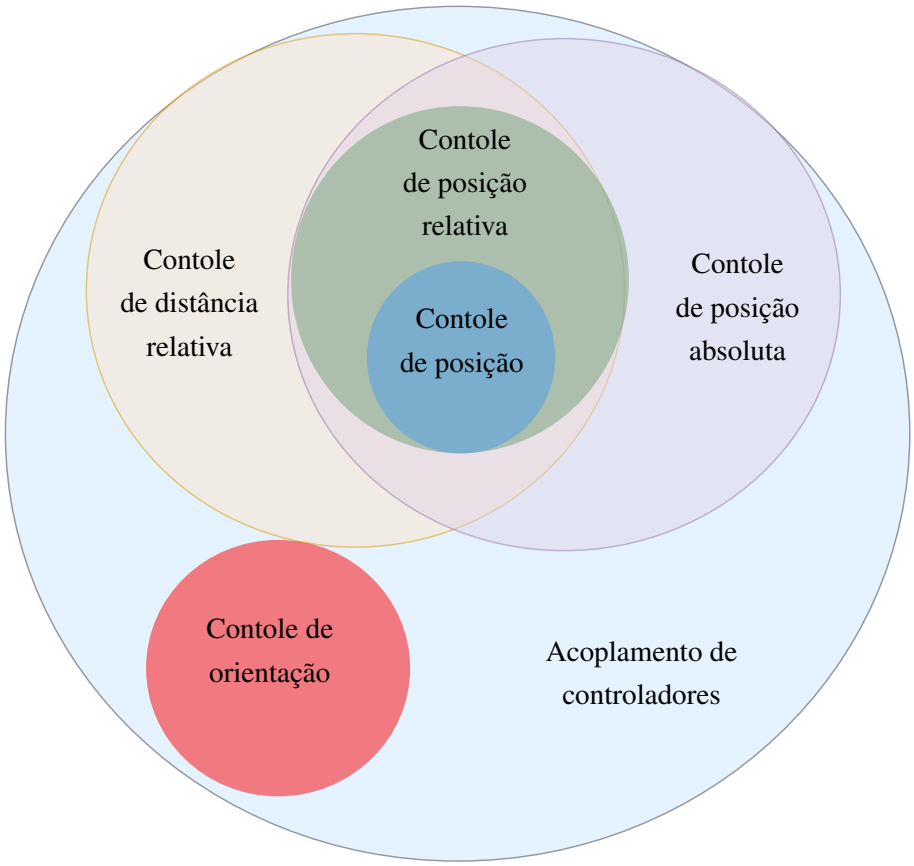


Fig. 6.4: Diagrama dos controladores projetados.

Além disso todos os controladores utilizaram ferramentas matemáticas como a Jacobiana definida por [2]

$$J = \left[vec_8 \underline{j_0 x} \dots vec_8 \underline{j_{n-1} x} \right], \tag{6.1}$$

em que

$$j_i = \underline{q}_0^i \underline{\varpi}_i^{i+1} \underline{q}_0^{i*} \tag{6.2}$$

e

$$\underline{\varpi}_i^{i+1} = \left[\left(-sen\alpha_i\hat{j} + cos\alpha_i\hat{k}\right) - a_i\left(cos\alpha_i\hat{j} + sen\alpha_i\right)\varepsilon\right]\dot{\theta}_i \tag{6.3}$$

em que a_i e α_i são parâmetros DH da junta i e q_o^i é definido a partir de (2.80).

6.4.1 Controle de Posição

O primeiro controlador implementado foi de translação pura, aqui se desejava deslocar o efetuador final de um dos braços para uma posição pré-definida, sem realizar um controle de orientação. Para completar a tarefa são necessários apenas três graus de liberdade, havendo, então, dois graus de redundância. Assim, inicialmente definiram-se as variáveis de referência, neste caso, t_{ref}, obtida a partir de (2.77).

Em seguida, deve-se calcular a Jacobiana de posição definida por [2]

$$J_p = 2\bar{H}_4(\mathcal{P}(\underline{q}^*))J_{\mathcal{D}(\underline{q})} + 2\overset{+}{H}_4(\mathcal{D}(\underline{q}))J^*_{\mathcal{P}(\underline{q})}, \tag{6.4}$$

em que $\underline{q}$ é o quatérnio dual unitário obtido através da cinemática direta do vetor de juntas e o conjugado da Jacobiana é dado por $J^*_{\mathcal{P}(\underline{q})} = C_4 J_{\mathcal{P}(\underline{q})}$, de modo que C_4 é calculado de maneira análoga a (2.56), sendo neste caso $C_4 = diag(1\ -1\ -1\ -1)$. Além disso definiu-se as Jacobianas como em (6.1) e os operadores de Hamilton como em (2.27) e (2.28).

Em seguida, se define o erro do sistema, neste caso, dado por

$$e_p = vec_4(t_{ref.} - t). \tag{6.5}$$

Aqui tem-se que t é translação de $\underline{q}$, calculada como em (2.77).

Dados o erro e a Jacobiana, já seria possível definir a lei de controle como em (2.95), entretanto, optou-se pela utilização do espaço nulo, como definido na Sec. 2.6.1. O espaço nulo, como já mencionado, tem a função de realizar tarefas secundárias, nesse caso evitar o limite de juntas. Desse modo, a ação de controle final pode ser dada por

$$\theta_{k+1} = \theta_k + J_p^\dagger K e_p + (J_s P_i)^\dagger \left(s - J_s J_p^\dagger K e_p\right), \tag{6.6}$$

em que K é um ganho positivo, s é uma função de mapeamento da distância entre ângulo da junta e o seu valor médio, dada por

$$s = \sum_{i=1}^{n} \frac{1}{2}\left(\theta_i - \overline{\theta}_i\right)^2 \tag{6.7}$$

e J_s é a Jacobiana de s, definida por

$$J_s = \nabla s = \left(\theta_1 - \overline{\theta}_1, \ldots, \theta_n - \overline{\theta}_n\right). \tag{6.8}$$

Nota-se que $\overline{\theta}_i$ é o valor médio da junta i. Desta maneira, no espaço nulo da tarefa principal, minimiza-se a relação entre as juntas e seus valores médios afastando-as de seus limites.

6.4.2 Controle de Orientação

Do mesmo modo que foi criado um controle para a translação pura, foi implementado um controlador puramente para a rotação. Aqui o objetivo foi fixar a atitude de um sistema de coordenadas.

Assim, primeiramente, foi definida uma orientação de referência, $\underline{q}_{ref}$, e em seguida fez-se o cálculo da Jacobiana de rotação, definida por

$$J_r = \bar{H}_4 \left(\underline{q}_{ref} \right) J^*_{\mathcal{P}(\underline{q})}. \tag{6.9}$$

Em seguida, calculou-se o erro do sistema, dado por

$$e_r = vec_4 \left(DQ\left(1\right) - \underline{q}^* \underline{q}_{ref} \right), \tag{6.10}$$

em que $DQ\left(1\right) = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$.

Finalmente aplicou-se o a lei de controle definida da forma

$$\theta_{k+1} = \theta_k + J_r^{\dagger} K e_r. \tag{6.11}$$

Aqui é possível ressaltar que não foi utilizado o espaço nulo, visto que para uma rotação do efetuador final não deverá haver grandes translações, tornando, portanto, tal mapeamento desnecessário.

6.4.3 Controle de Posição Relativa no Espaço Cooperativo

Para este controlador, buscou-se criar uma implementação que controlasse a posição de um braço em relação a outro, desse modo introduziu-se o vetor de juntas aumentado $\Theta = \left[\theta_{Esq}^T\ \theta_{Dir}^T\right]^T$ e os conceitos de espaço cooperativo dual como definidos na Sec. 2.7.1. De modo análogo ao controle de posição realizado anteriormente (Sec. 6.4.1), foram calculados os quatérnios duais unitários para os efetuadores finais de cada braço, $\underline{q}_{Esq}$e $\underline{q}_{Dir}$.

O controle foi realizado com base na posição relativa do braço esquerdo em relação ao direito e, para tanto utilizou-se (2.99), sendo $\underline{q}_{rel} = \underline{q}^*_{Dir}\underline{q}_{Esq}$. Desse modo, foi feito o cálculo da Jacobiana relativa dada por [2]

$$J_{rel} = \left[\overset{+}{H} \left(\underline{q}^*_{Dir} \right) J_{Esq}\ \bar{H} \left(\underline{q}_{Esq} \right) J^*_{Dir} \right], \tag{6.12}$$

em que J_{Esq} e J_{Dir} são as Jacobianas analíticas como definido em (6.1) e o conjugado da Jacobiana é dado de forma análoga a Jacobiana de posição, de modo que $J^*_{Dir} = C_8 J_{Dir}$, sendo C_8 definido em (2.56).

Como a tarefa definida é o controle de posição relativa, houve a necessidade de se calcular também a Jacobiana de posição $J_{P_{rel}}$, da mesma maneira que em (6.4), apenas substituindo J por J_{rel}. Então calculou-se o erro como

$$e_{P_{rel}} = vec_4(t_{ref.} - t_{rel}), \tag{6.13}$$

sendo t_{ref} uma posição de referência e t_{rel} a translação relativa obtida de $\underline{q}_{rel}$ com (2.77).

Com a Jacobiana e o erro calculados foi possível definir uma lei de controle, definida por

$$\Theta_{k+1} = \Theta_k + J^{\dagger}_{P_{rel}} K e_{P_{rel}} + \left(J_{s_{Aug}} P_{i_{Aug}}\right)^{\dagger} \left(s_{Aug} - J_{s_{Aug}} J^{\dagger}_{P_{rel}} K e_{P_{rel}}\right), \tag{6.14}$$

em que K é uma matriz diagonal de ganhos e $J^{\dagger}_{P_{rel}}$ é a pseudoinversa de $J_{P_{rel}}$ definida em (6.12).

Utilizou-se o espaço nulo aumentado, de modo que s_{Aug}, $P_{i_{Aug}}$ e $J_{s_{Aug}}$ são a tarefa secundária, a projeção do espaço nulo da tarefa principal e a Jacobiana da tarefa secundária, os quais são definidos por

$$s_{Aug} = s_{Esq} + s_{Dir}, \tag{6.15}$$

$$P_{i_{Aug}} = \left(I_{10} - J^{\dagger}_{rel} J_{rel}\right), \tag{6.16}$$

e

$$J_{Aug} = \left[J_{s_{Esq}}\ J_{s_{Dir}}\right], \tag{6.17}$$

em que s_{Esq} e s_{Dir} são obtidos de forma análoga a (6.7) e $J_{s_{Esq}}$ e $J_{s_{Dir}}$ como em (6.8).

6.4.4 Controle de Distância Relativa no Espaço Cooperativo

O controle de posição relativa tinha como intuito definir uma translação de um braço em relação a outro, entretanto essa operação utilizava três graus de liberdade. Assim, foi proposto um controlador que atuasse para manter uma distância fixa em um eixo entre os braços. Definiu-se o vetor de juntas aumentado $\Theta = \left[\theta^T\ \theta^T\right]$ e, a partir da cinemática direta, os quatérnios duais unitários para os efetuadores de cada braço $\underline{q}_{Esq}$e $\underline{q}_{Dir}$. Além disso, foram calculados $\underline{q}_{rel}$ e J_{rel} como na Sec. 6.4.3 e com estes pode-se obter a Jacobiana de distância, definida por [2]

$$J_{D_{rel}} = 2\left(vec_4(t_{rel})\right)^T J_{P_{rel}}. \tag{6.18}$$

Nota-se que t_{rel} é a translação obtida de $\underline{x}_{rel}$ como mostrado em (2.77).

Em seguida, foi calculado o erro, definido da forma

$$e_{D_{rel}} = d - \left\|vec_4(t_{rel})\right\|, \tag{6.19}$$

em que se definiu d como uma distância de referência em metros.

Finalmente, definiu-se a ação de controle como

$$\Theta_{k+1} = \Theta_k + J^{\dagger}_{D_{rel}} K e_{D_{rel}} + \left(J_{s_{Aug}} P_{i_{Aug}}\right)^{\dagger} \left(s_{Aug} - J_{s_{Aug}} J^{\dagger}_{D_{rel}} K e_{D_{rel}}\right), \tag{6.20}$$

em que o espaço nulo é definido de maneira análoga ao que foi implementado na Sec. 6.4.3.

6.4.5 Controle de Posição Absoluta no Espaço Cooperativo

A posição absoluta é dada pelo ponto médio entre os braços, como foi definido na Sec.2.7.1. Assim, mais uma vez se utilizando o vetor aumentado de juntas $\Theta = \begin{bmatrix}\theta^T \ \theta^T\end{bmatrix}$ e os quatérnios duais unitários dos efetuadores dos braços $\underline{q}_{Esq}$e $\underline{q}_{Dir}$, calculou-se q_{rel}, como na Sec. 6.4.3 além de $\underline{q}_{rel/2}$ e $\underline{q}_{abs}$ como definido na Sec. 2.7.1. Tem-se que $q_{rel/2}$ e q_{abs} podem ser definidos como

$$\underline{q}_{rel/2} = exp\left(\frac{log(\underline{q}_{rel})}{2}\right) \tag{6.21}$$

e

$$\underline{q}_{abs} = \underline{q}_{Dir}\underline{q}_{rel/2}. \tag{6.22}$$

Assim foi possível calcular as Jacobianas $J_{rel/2}$ e J_{abs} como [2]

$$J_{rel/2} = \begin{bmatrix} \frac{1}{2}\left(\bar{H}_4(\mathcal{P}(\underline{q}^*_{rel/2}))J_{\mathcal{P}(\underline{q}_{rel})}\right) \\ \frac{1}{4}\left(\bar{H}_4(\mathcal{P}(\underline{q}_{rel/2}))J_{P_{rel}} + \overset{+}{H}_4(t_{rel})J_{\mathcal{P}\left(\underline{q}_{rel/2}\right)}\right) \end{bmatrix} \tag{6.23}$$

e

$$J_{abs} = \bar{H}\left(\underline{q}_{rel/2}\right)J_{Dir_{ext}} + \overset{+}{H}\left(\underline{q}_{Dir}\right)J_{rel/2}, \tag{6.24}$$

respectivamente, em que se define $J_{Dir_{ext}}$ como

$$J_{Dir_{ext}} = \begin{bmatrix}0_{8\times n_{Esq}} \ J_{Dir}\end{bmatrix}, \tag{6.25}$$

sendo n_{Esq} o número de juntas do braço esquerdo.

Com a definição de J_{abs}, pode-se utilizar (6.4), apenas substituindo J por J_{abs}, de modo a se obter $J_{P_{abs}}$.

Finalmente, calculou-se o erro

$$e_{P_{abs}} = vec_4(t_{ref.} - t_{abs}), \tag{6.26}$$

em que t_{ref} é uma posição de referência e t_{abs} a translação de $\underline{q}_{abs}$.

A ação de controle, pode, então, ser dada por

$$\Theta_{k+1} = \Theta_k + J^{\dagger}_{P_{abs}}Ke_{P_{abs}} + \left(J_{s_{Aug}}P_{i_{Aug}}\right)^{\dagger}\left(s_{Aug} - J_{s_{Aug}}J^{\dagger}_{P_{abs}}Ke_{P_{abs}}\right), \tag{6.27}$$

em que K é uma matriz de ganhos e o espaço nulo é definido como na Sec. 6.4.3.

Fig. 6.5: Exemplos

6.4.6 Concatenação de Controladores

Cada um dos controladores projetados nas Secs. 6.4.3 a 6.4.5 requerem diferentes graus de liberdade, desse modo, definiu-se a Tab. 6.5 [6].

Tab. 6.5: Graus de liberdade requeridos para realização de diferentes tarefas.

	Tarefa	DoFs Requeridos
Manipulador Único	Posição	3
	Orientação	3
	Pose	6
Espaço Cooperativo	Distância Relativa	1
	Posição Relativa	3
	Orientação Relativa	3
	Pose Relativa	6
	Posição Absoluta	3
	Orientação Absoluta	3
	Posição Completa	6
	Orientação Completa	6
	Pose Completa	12

Imaginando, então, uma tarefa em que um robô carregaria um objeto, por exemplo uma bandeja, haveria a necessidade de se controlar mais de uma das variáveis definidas na Tab. 6.5. Desse modo, tomando como base a plataforma NAO com dez graus de liberdade foi projetado um controlador com as seguintes restrições:

- imaginando duas alças verticais, definiu-se que os efetuadores de cada braço do robô deveriam estar alinhados e espelhados. Desse modo, utilizando o controle de orientação de cada braço necessitando, portanto, de seis graus de liberdade;
- para se carregar objetos rígidos o robô deverá manter a distância entre seus braços constante, assim implementou-se o controle de distância relativa que necessita apenas de um grau de liberdade;
- afim se se movimentar um objeto no espaço, utilizou-se um controlador de posição absoluta, necessitando de três graus de liberdade.

Dadas as três tarefas propostas, concatenou-se os controladores de orientação, distância relativa e posição absoluta definidos respectivamente nas Secs. 6.4.2, 6.4.4 e 6.4.5.

Assim a Jacobiana pode ser definida como a concatenação das Jacobianas individuais de cada controlador, dada por

$$J_C = \left[\left[J_{r_{Esq}} \; 0_{4\times n_{Esq}} \right]^T \; \left[0_{4\times n_{Dir}} \; J_{r_{Dir}} \right]^T \; \left[J_{D_{rel}} \right]^T \; \left[J_{P_{abs}} \right]^T \right]^T \tag{6.28}$$

e o erro é definido pelo vetor aumentado

$$e_C = \left[\left[e_{r_{Esq}} \right]^T \; \left[e_{r_{Dir}} \right]^T \; \left[e_{D_{rel}} \right]^T \; \left[e_{P_{abs}} \right]^T \right]^T . \tag{6.29}$$

Finalmente, define-se a ação de controle como

$$\theta_{k+1} = \theta_k + J_C^{\dagger} K e_C, \tag{6.30}$$

em que não se utilizou o espaço nulo dado que todos os graus de liberdade do robô estavam sendo utilizados pelo controlador.

6.4.7 Resumo dos Controladores Implementados

Dados os diversos controladores implementados, é interessante compará-los quanto suas variáveis de controle, Jacobianas e erros utilizados além de quantos graus de liberdade a tarefa necessita e se utiliza o espaço nulo. A Tab. 6.6 (página 75) mostra tal comparação.

Tab. 6.6: Comparação entre controladores

	Variável	Jacobiana	Erro	DoF	Espaço Cooperativo	Espaço Nulo
Controle de Posição	$t = \begin{cases} x \\ y \\ z \end{cases}$	$J_p = 2\bar{H}_4(\mathcal{P}(\underline{q}^*))J_{\mathcal{D}(\underline{q})} + 2\overset{+}{H}_4(\mathcal{D}(\underline{q}))J^*_{\mathcal{P}(\underline{q})}$	$e_p = vec_4(t_{ref.} - t)$	3	Não	Sim
Controle de Orientação	$r = \begin{cases} cos\left(\frac{\phi}{2}\right) + sen\left(\frac{\phi}{2}\right) i \\ cos\left(\frac{\phi}{2}\right) + sen\left(\frac{\phi}{2}\right) j \\ cos\left(\frac{\phi}{2}\right) + sen\left(\frac{\phi}{2}\right) k \end{cases}$	$J_r = \bar{H}_4\left(\underline{q}_{ref}\right) J^*_{\mathcal{P}(\underline{q})}$	$e_r = vec_4\left(DQ\left(1\right) - \underline{q}^*\underline{q}_{ref}\right)$	3	Não	Não
Controle de Posição Relativa	$\underline{q}_{rel} = \underline{q}^*_{Dir}\underline{q}_{Esq}$	$J_{rel} = \left[\overset{+}{H}\left(\underline{q}^*_{Dir}\right) J_{Esq}\ \bar{H}\left(\underline{q}_{Esq}\right) J^*_{Dir}\right]$	$e_{P_{rel}} = vec_4(t_{ref.} - t_{rel})$	3	Sim	Sim
Controle de Distância Relativa	$d_{rel} = \left\|\underline{q}_{rel}\right\|$	$J_{D_{rel}} = 2\left(vec_4(t_{rel})\right)^T J_{P_{rel}}$	$e_{D_{rel}} = d - \|vec_4(t_{rel})\|$	1	Sim	Sim
Controle de Posição Absoluta	$\underline{q}_{\frac{rel}{2}} = exp\left(\frac{log(\underline{q}_{rel})}{2}\right)$ $\underline{q}_{abs} = \underline{q}_{Dir}\underline{q}_{\frac{rel}{2}}$	$J_{rel/2} = \begin{bmatrix} \frac{1}{2}\left(\bar{H}_4(\mathcal{P}(\underline{q}^*_{rel/2}))J_{\mathcal{P}(\underline{q}_{rel})}\right) \\ \frac{1}{4}\left(\bar{H}_4(\mathcal{P}(\underline{q}_{rel/2}))J_{P_{rel}} + \overset{+}{H}_4(t_{rel})J_{\mathcal{P}\left(\underline{q}_{rel/2}\right)}\right) \end{bmatrix}$ $J_{abs} = \bar{H}\left(\underline{q}_{rel/2}\right) J_{Dir_{ext}} + \overset{+}{H}\left(\underline{q}_{Dir}\right) J_{rel/2}$	$e_{P_{abs}} = vec_4(t_{ref.} - t_{abs})$	3	Sim	Sim
Concatenação de Controladores	$r_{Esq}, r_{Dir}, d_{rel}, \underline{q}_{abs}$	$J_C = \begin{bmatrix} \left[J_{r_{Esq}}\ 0_{4\times n_{Esq}}\right]^T \\ \left[0_{4\times n_{Dir}}\ J_{r_{Dir}}\right]^T \\ \left[J_{D_{rel}}\right]^T \\ \left[J_{P_{abs}}\right]^T \end{bmatrix}$	$e_C = \begin{bmatrix} \left[e_{r_{Esq}}\right]^T \\ \left[e_{r_{Dir}}\right]^T \\ \left[e_{D_{rel}}\right]^T \\ \left[e_{P_{abs}}\right]^T \end{bmatrix}$	10	Sim	Não

7 RESULTADOS: CONTROLE COOPERATIVO

"A straight line may be
the shortest distance between two points, but it is by no means the most interesting."

The Doctor

7.1 INTRODUÇÃO

Nesta seção serão apresentados os resultados referentes ao sistema de controle com base na matemática dos quatérnios duais e espaço cooperativo dual. Aqui primeiramente é feita uma análise do comportamento do sistema estruturado com o ROS e implementado na plataforma NAO e, em seguida, são analisados os resultados obtidos a partir dos diversos controladores implementados, tanto em simulação quanto no robô real.

7.2 TESTES DE SENSORES E ATUADORES

Visando analisar o funcionamento do sistema implementado na plataforma física, obter dados como o tempo de resposta e checar se os dados lidos e enviados para os atuadores estavam corretos, foi feita uma simplificação do sistema mostrado na Fig. 6.3. De modo que os blocos de controle foram removidos e os dados obtidos no bloco de leitura foram mandados diretamente para o bloco de atuação.

Para testar o sistema proposto foi realizado um experimento em que se desligou os motores de um dos braços do NAO e se fez a leitura de sua posição. Esta posição era espelhada e mandada para o outro braço, criando um sistema mestre-escravo. Foi feita então uma movimentação manual de um dos braços e o outro realizou o movimento correspondente. Os resultados deste experimento podem ser observados nos gráficos da Fig. 7.1.

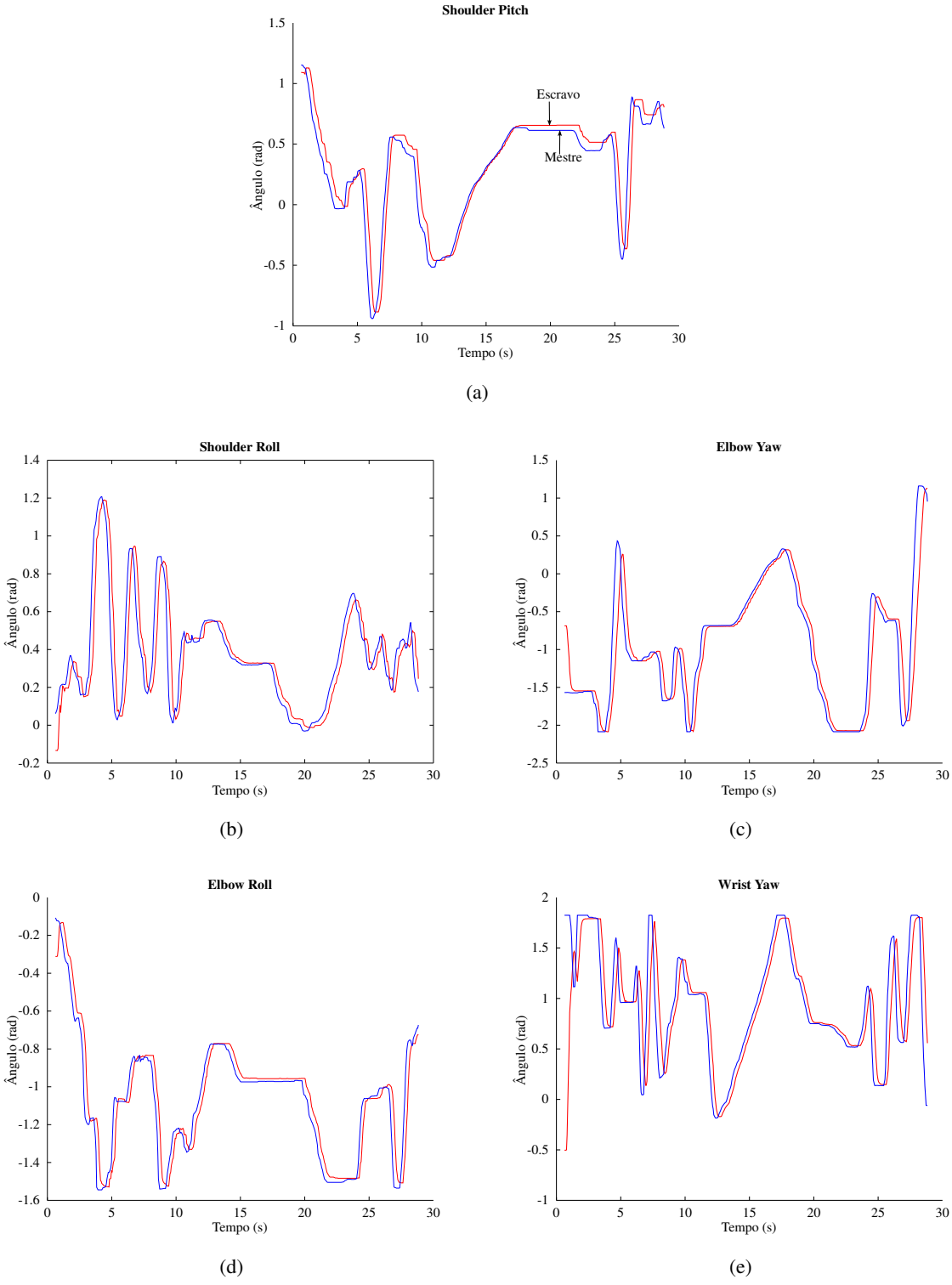


Fig. 7.1: Gráficos de Movimento das Juntas Espelhadas

Nota-se que entre a curva do mestre e do escravo há uma pequena diferença: o escravo encontra-se deslocado no tempo. A partir disso, pode-se calcular o tempo médio de resposta do sistema como sendo 266.66 ms. Ainda pode-se notar que o sistema responde bem ao comando para atuação, visto que para valores de ângulo variáveis a o seu erro médio se manteve baixo, em geral com valores menores que 0.0390 radianos (2.235^o). A Tab. 7.1 mostra os valores de erro médio obtidos para cada

junta do robô.

Tab. 7.1: Erro médio entre a posição lida e atuada em cada junta do robô

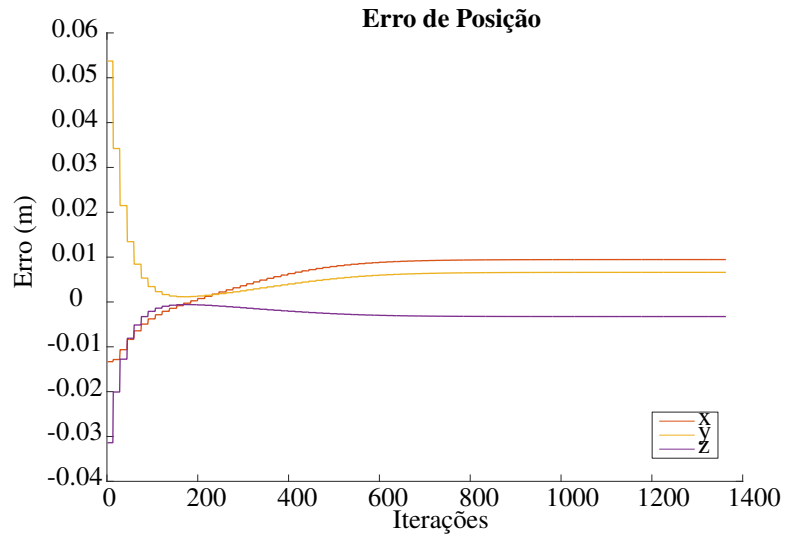
Junta	Erro Médio (rad)
Shoulder Pitch	0.0390
Shoulder Roll	0.0002
Elbow Yaw	0.0110
Elbow Row	0.0105
Wrist Yaw	0.0123

7.3 CONTROLE DE POSIÇÃO

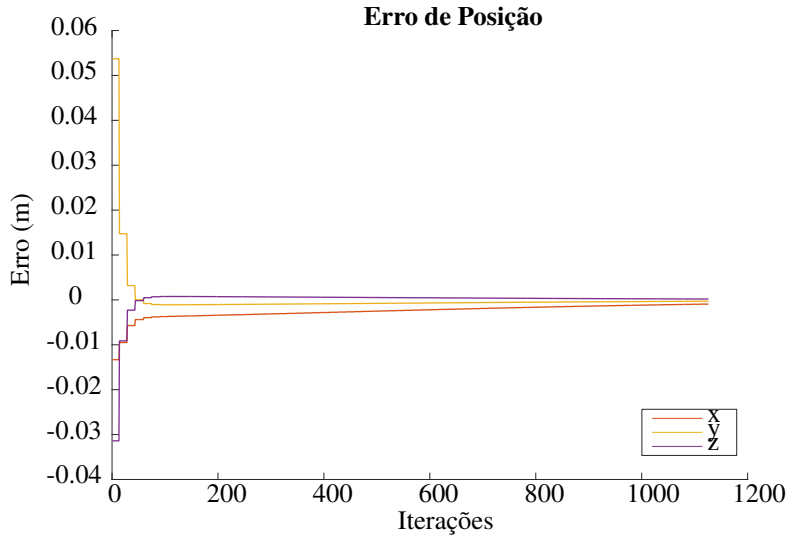
O controlador de posição, que aplica uma translação em um dos braços, foi o primeiro a ser implementado com quatérnios duais na plataforma NAO. Como descrito na Sec. 6.4.1, foram calculadas a Jacobiana, o erro e a lei de controle foi aplicada com o espaço nulo. Desse modo, pode-se observar tanto os resultados deste controlador em um robô simulado quanto em uma plataforma física.

7.3.1 Simulação

Para verificar o correto funcionamento do controlador de posição implementado e evitar causar danos ao robô real, primeiramente foram realizadas simulações no ambiente V-Rep. Os resultados do controle de posição do braço esquerdo podem ser observados na Fig. 7.2.



(a) Braço esquerdo com controle no espaço nulo



(b) Braço esquerdo sem controle no espaço nulo

Fig. 7.2: Controlador de Posição do Braço Esquerdo em robô simulado no V-Rep.

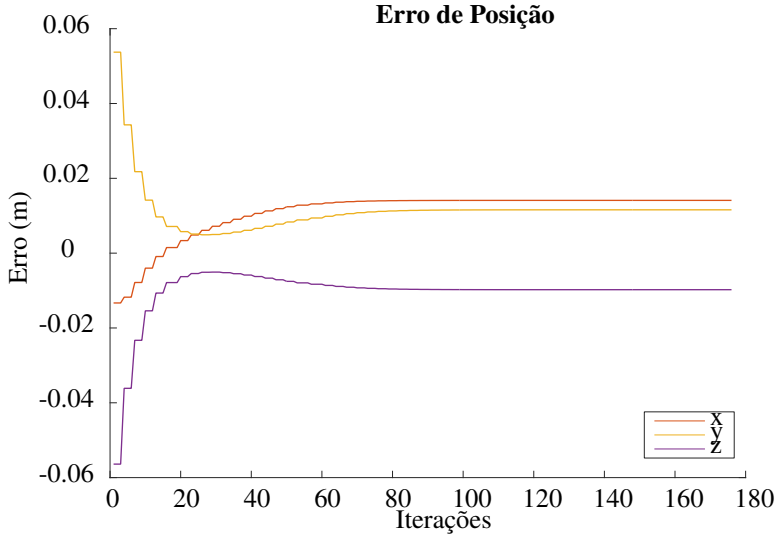
Nota-se que os gráficos mostram o erro da posição do efetuador em relação ao objetivo, sendo a Fig. 7.2a os resultados obtidos para o controlador com espaço nulo e a Fig. 7.2b sem utilizar tal espaço.

Primeiramente analisa-se que o controlador com espaço nulo apresenta um erro mais alto, e que ele deixa de convergir para zero em um dado momento, se estabilizando em um erro próximo de 0.01m enquanto a mesma tarefa sendo executada sem o espaço nulo obteve um erro no estado estacionário de 0.0009m. Inicialmente cogitou-se que tal erro fosse devido a dinâmica do sistema, dado que a simulação do V-Rep leva em conta aspectos da física do mundo real. De fato, sabendo que o espaço nulo, definido pela Jacobiana, garante que nenhuma velocidade será aplicada no espaço de trabalho, não há garantia, entretanto, de que as forças agindo no sistema irão gerar também resultados para o espaço nulo da aceleração [42]. Afim de se verificar esta hipótese, executou-se a simulação numérica

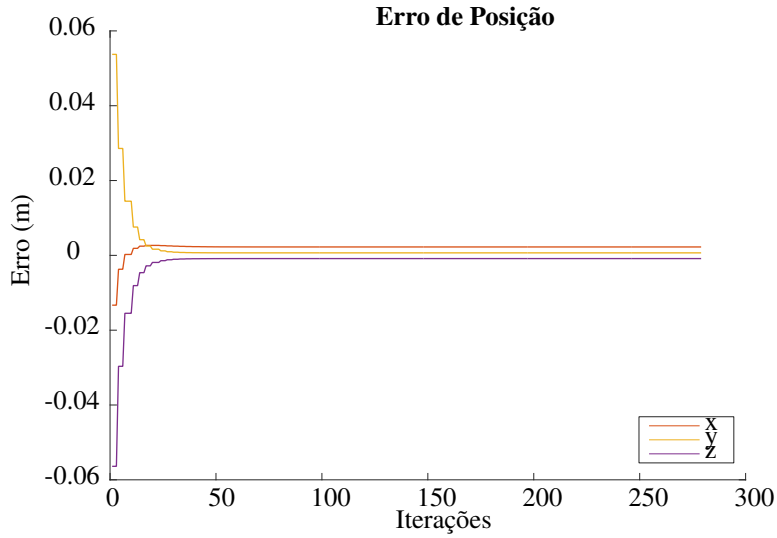
com o espaço nulo, utilizando apenas o Matlab que não foi configurado para considerar a dinâmica do sistema. Verificou-se, entretanto, que o comportamento do controlador continuava o mesmo, logo tal diferença não foi devida a diferenças dinâmicas.

Ainda com o intuito de explicar o erro mais alto, fez-se o calculo dos autovalores da multiplicação entre a Jacobiana da tarefa com o projetor do espaço nulo, tal multiplicação mostra a interferência do espaço nulo no espaço da tarefa. Idealmente este resultado deve estar próximo de zero, entretanto, verificou-se que dois dos quatro autovalores eram não nulos. De fato, verifica-se no gráfico da Fig. 7.2a que dois dos três erros são mais afetados.

Foi feita, então, uma nova análise do controlador, dessa vez verificando o fator de amortecimento. O fator de amortecimento foi reduzido em 100 vezes, de modo a verificar o comportamento do espaço nulo resultante. Observa-se na Fig. 7.3, que o espaço nulo deixou de afetar o erro na tarefa do controlador principal a partir do momento que houve a redução do fator de amortecimento.



(a) Fator de Amortecimento de 10^{-3}



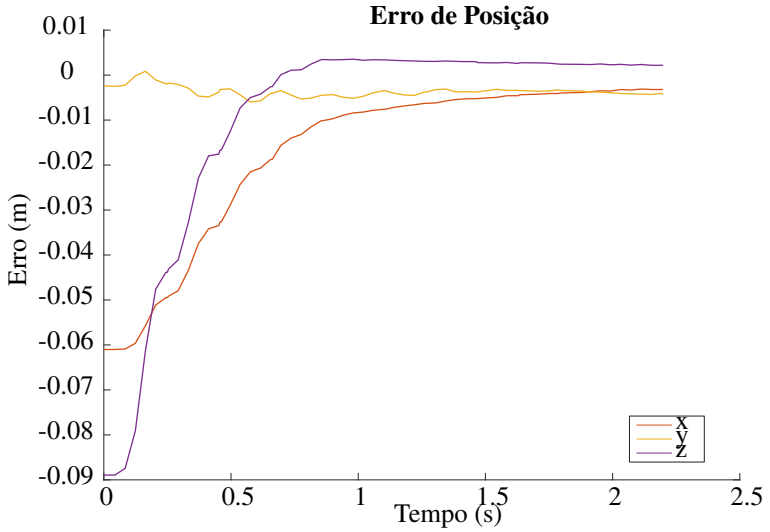
(b) Fator de Amortecimento de 10^{-5}

Fig. 7.3: Efeito do fator de amortecimento no comportamento do espaço nulo

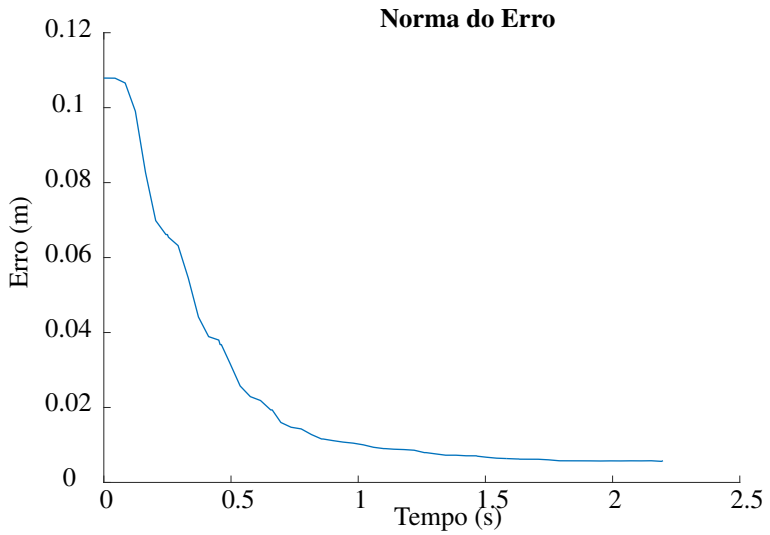
Tal análise é decorrente da propagação de erro na velocidade da trajetória, inclusive no espaço nulo, gerada pelo fator de amortecimento.

7.3.2 Plataforma Física

Dado que o robô simulado realizou a tarefa com sucesso, foi feito o mesmo experimento com o robô físico, também utilizando o espaço nulo. Aqui obteve-se os resultados vistos na Fig. 7.4. Primeiramente, pode-se observar na Fig. 7.4b que a norma do erro se estabilizou em 5.7mm com o tempo de assentamento próximo de 1 segundo. Além disso o fator de amortecimento utilizado foi de 10^{-3}.



(a) Erro do Braço Esquerdo implementado em robô real



(b) Norma do erro da posição do braço esquerdo

Fig. 7.4: Trajetória e Orientação do Efetuador

Observa-se, então, o erro do controlador. Dado que foi implementado um controlador propor-

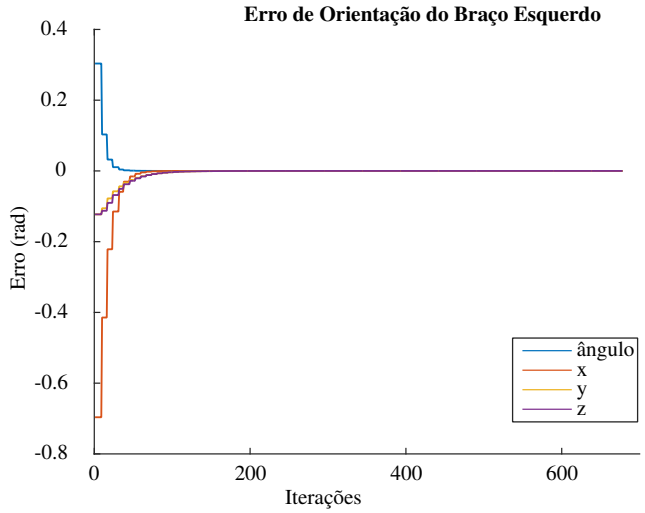
cional, em que um ganho multiplica o erro, era esperado que o erro não seria nulo.

7.4 CONTROLE DE ORIENTAÇÃO

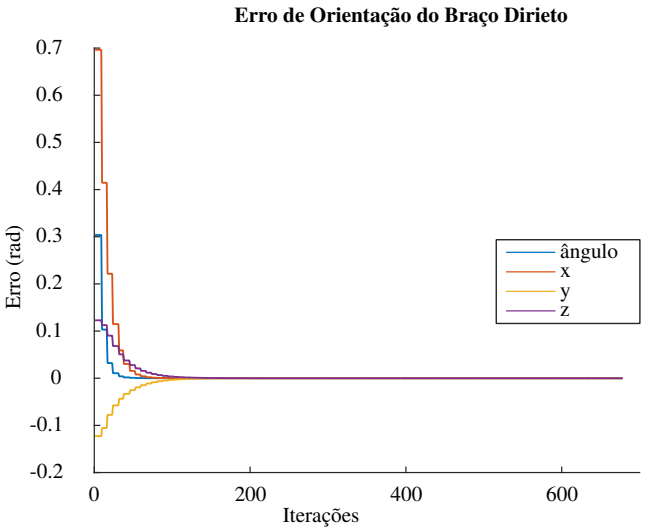
O controle de orientação buscava manter fixa a orientação de ambos os braços em relação a base do robô. Desse modo, implementou-se um controlador como descrito na Sec. 6.4.2 em cada braço, espelhando a orientação de um efetuador em relação ao outro.

7.4.1 Simulação

Os resultados simulados para a rotação dos braços em simulação podem ser observados na Fig. 7.5.



(a) Erro do braço esquerdo



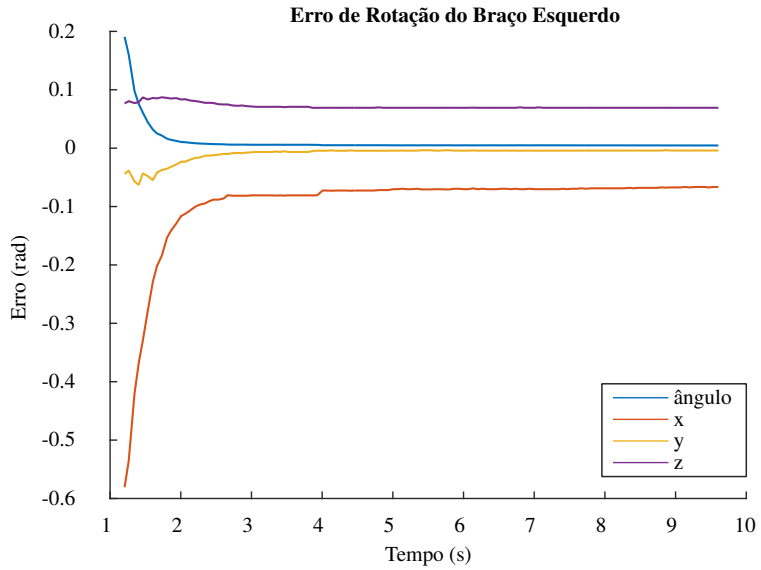
(b) Erro do braço direito

Fig. 7.5: Erro de orientação do efetuador

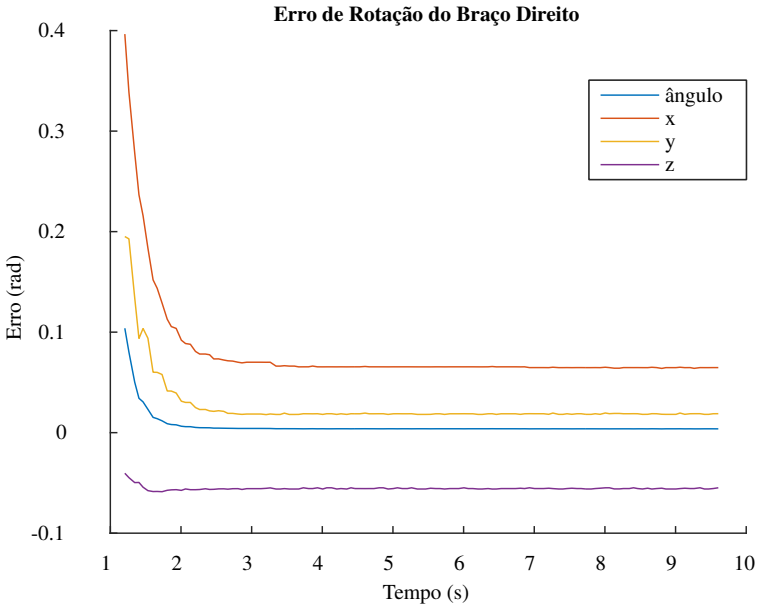
Aqui se observa que a orientação era dada a partir de uma rotação em torno de apenas um dos eixos da base, o que pode explicar a rápida convergência. Além disso o erro se encontra na casa de 10^{-10}, o que pode ser considerado praticamente nulo.

7.4.2 Plataforma Física

Para a plataforma física tem-se os resultados da Fig. 7.6. Diferente do controlador implementado em simulação, este possui erros maiores, dessa vez na ordem de 10^{-2}. Tal diferença é dada pelo fato do robô real possuir não linearidades no sistema e cada junta possuir uma zona morta devido a sua estrutura mecânica.



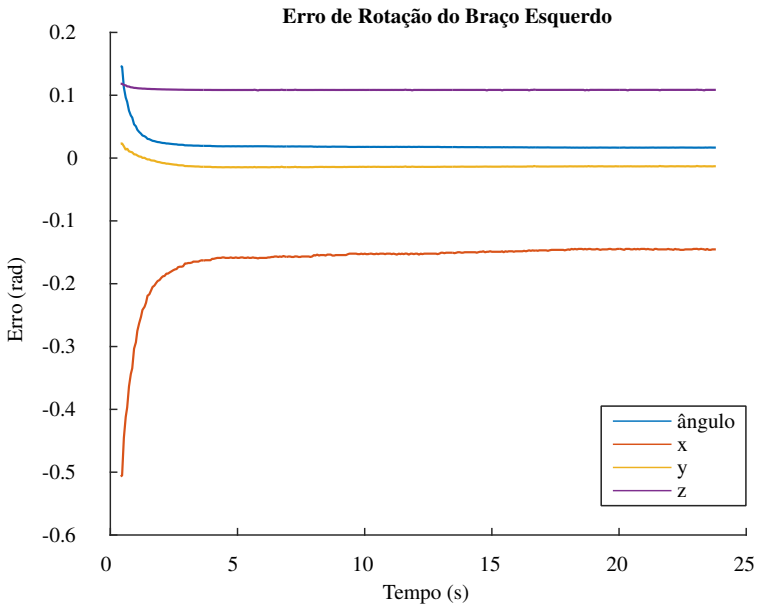
(a) Erro do braço esquerdo



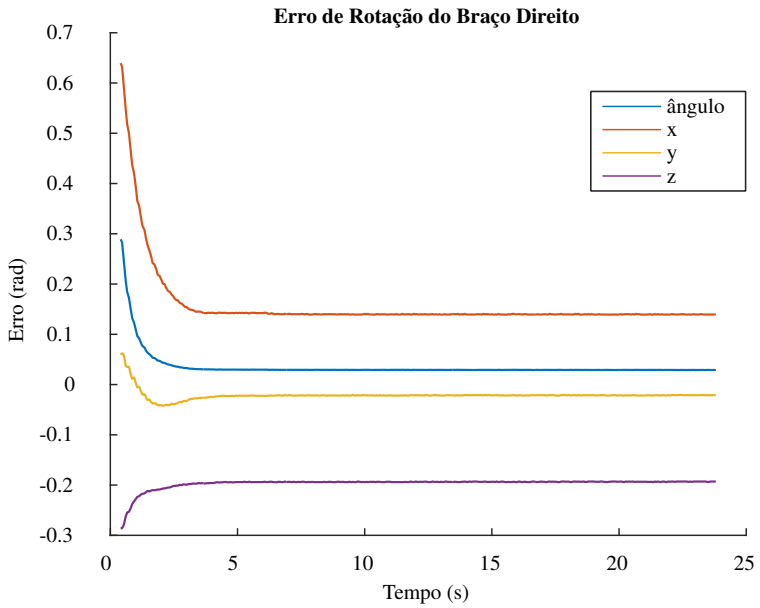
(b) Erro do braço direito

Fig. 7.6: Erros de orientação para cada braço

Observa-se que a Jacobiana faz seu mapeamento minimizando os esforços das juntas por mínimos quadrados a cada iteração, desse modo, a minimização é local, o que não garante uma minimização global. Isso pode ser verificado com a análise da Fig. 7.7, em que se observa os resultados do mesmo controlador com mesma referência operando a partir de uma posição inicial diferente. Nota-se que para este caso o erro é maior, o que indica que as minimizações locais não foram as melhores possíveis em um aspecto global.



(a) Erro do braço esquerdo



(b) Erro do braço direito

Fig. 7.7: Erros de orientação para cada braço para posição inicial alterada

Na Fig. 7.8 tem-se as orientações e posições iniciais e finais de cada efetuador para ambos os testes.

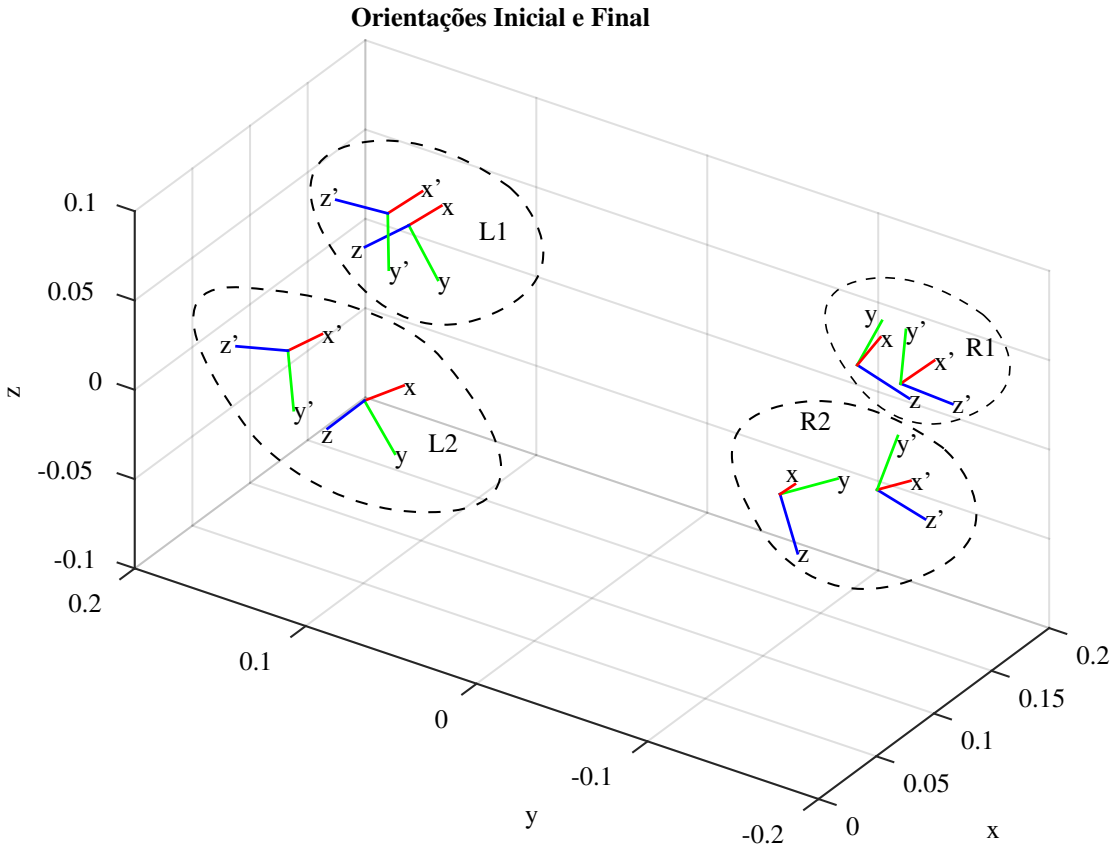


Fig. 7.8: Posição inicial e final dos efetuadores para ambos os braços. L1 e R1 são referentes ao teste realizado na Fig. 7.6 e L2 e R2 são referentes ao teste realizado na Fig. 7.7. Os sistemas denotados por $[x\ y\ z]$ são referentes às posições iniciais, enquanto $[x'\ y'\ z']$ se referem às posições finais.

7.5 CONTROLE DE POSIÇÃO RELATIVA NO ESPAÇO COOPERATIVO

O controle de posição relativa, como descrito na Sec. 6.4.3, tem como objetivo transladar um braço em relação a outro, de modo que possa se manter uma posição relativa fixa entre ambos. Desse modo, implementou-se o controlador tanto em simulação quanto na plataforma real.

7.5.1 Simulação

Para a verificação do controlador simulado, analisou-se a Fig. 7.9 que mostra o erro da posição do efetuador em relação a posição de referência. Aqui observa-se a convergência do controlador com o erro se estabilizando em torno de 10^{-4}.

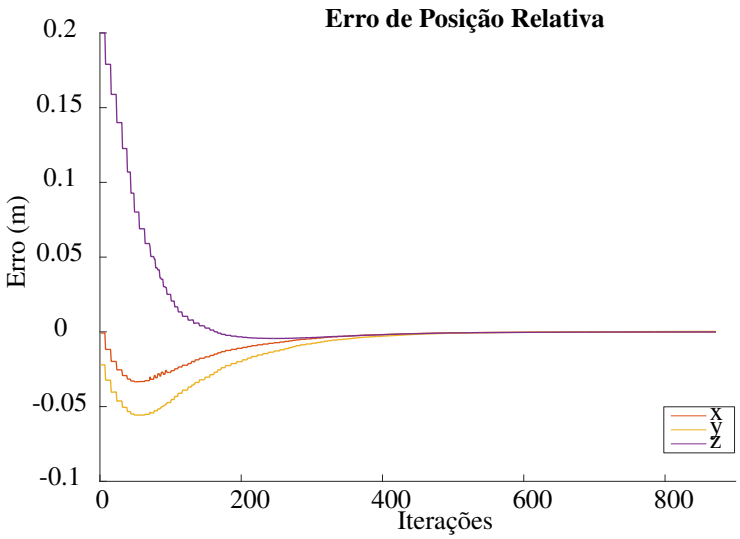
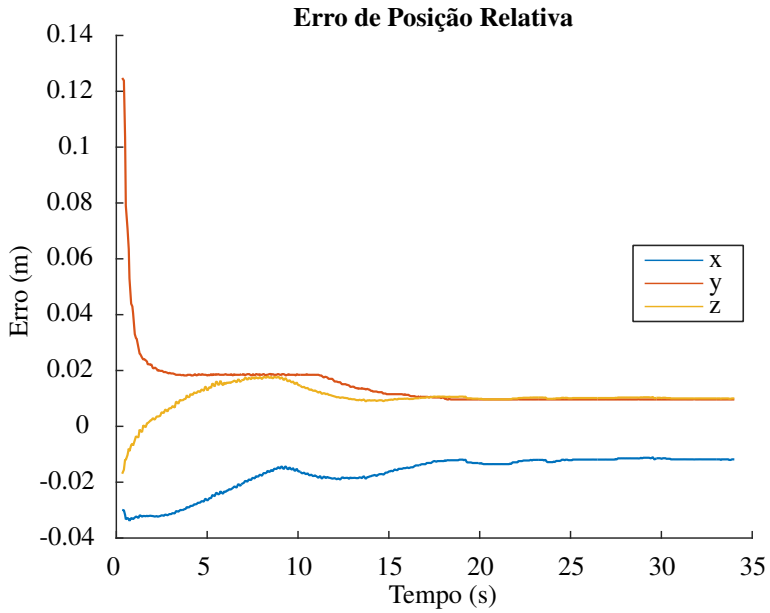


Fig. 7.9: Controlador de Posição relativa do braço esquerdo em relação ao direito

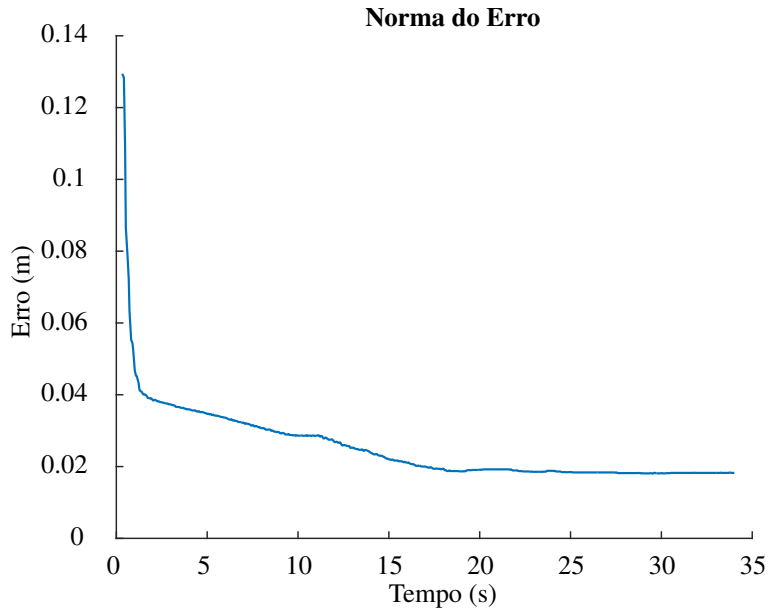
Como esperado, a simulação não sofre perturbações como no mundo real, logo o controlador consegue convergir com uma velocidade alta e manter o erro baixo.

7.5.2 Plataforma Física

O controlador aplicado na plataforma real é mostrado na Fig. 7.10. A primeira análise a ser feita em tal controlador é a do seu erro, que se estabiliza próximo de 2cm.



(a) Erro da posição relativa



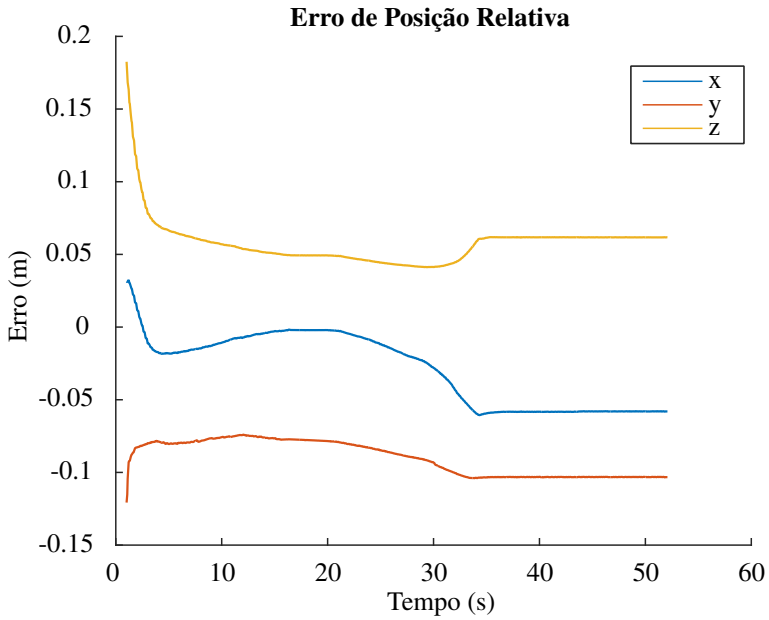
(b) Norma do erro da posição relativa

Fig. 7.10: Erro do controlador de posição relativa

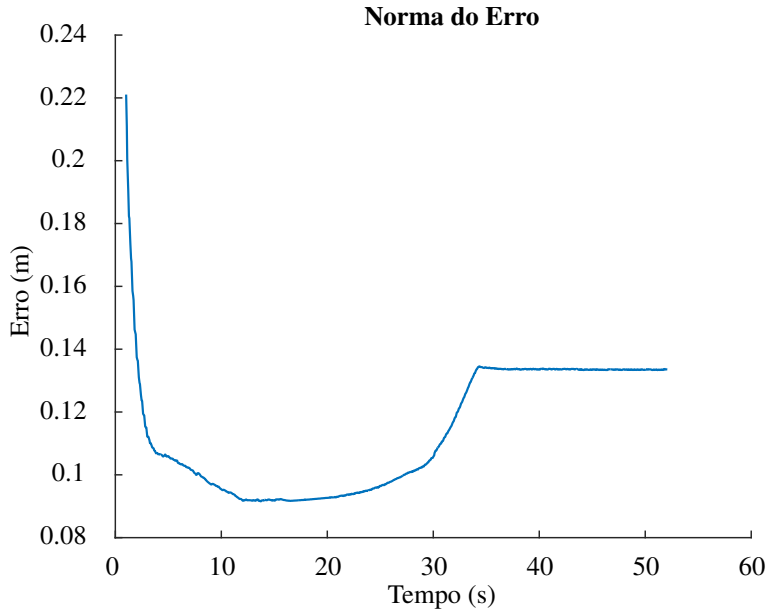
Comparando-se com os resultados simulados, verifica-se que o erro que era quase nulo passou a ser significativo, o que é um resultado esperado quando se leva em conta as folgas do robô real e as perturbações que este pode sofrer.

Além disso, nota-se que uma desvantagem do espaço cooperativo é a diminuição do espaço de trabalho do robô. De fato, considerando uma construção antropomórfica — em particular de um humanoide — é intuitivo se pensar que haverão posições alcançadas por um braço mas que o outro não chegará, dadas as restrições mecânicas introduzidas no modelo. Assim pode-se observar a Fig. 7.11 em que é mostrado o erro quando o controlador tenta mandar um dos braços para uma posição

inalcançável.



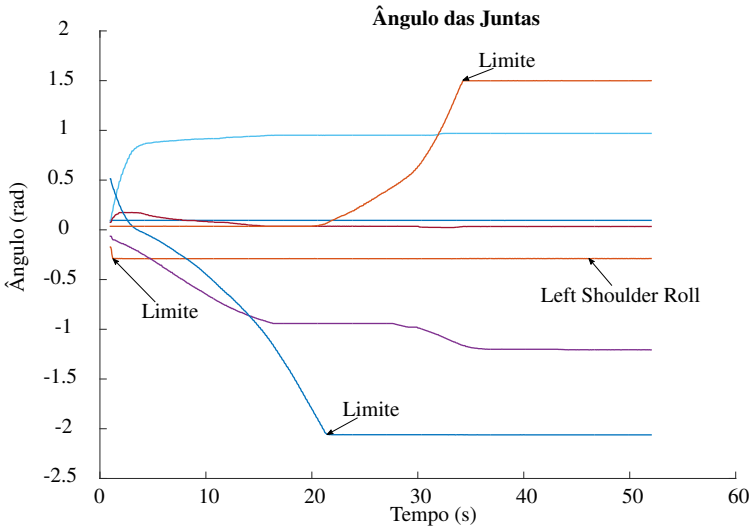
(a) Erro da posição relativa



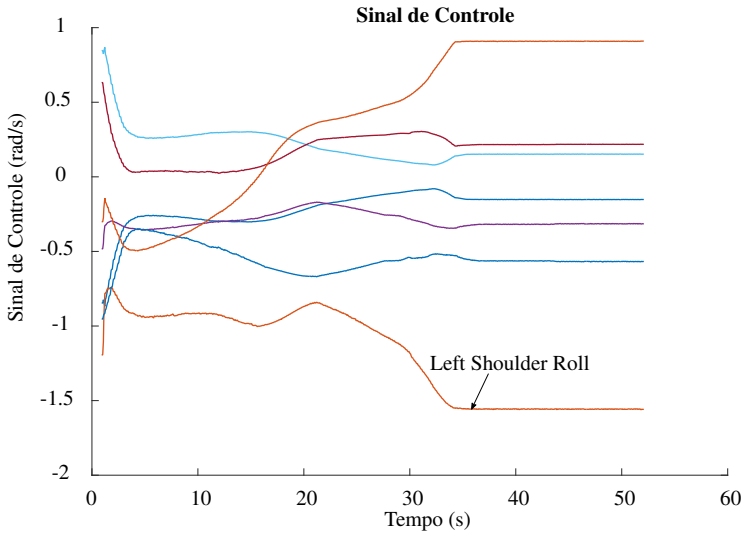
(b) Norma do erro da posição relativa

Fig. 7.11: Erro do controlador de posição relativa com limitações no espaço de trabalho

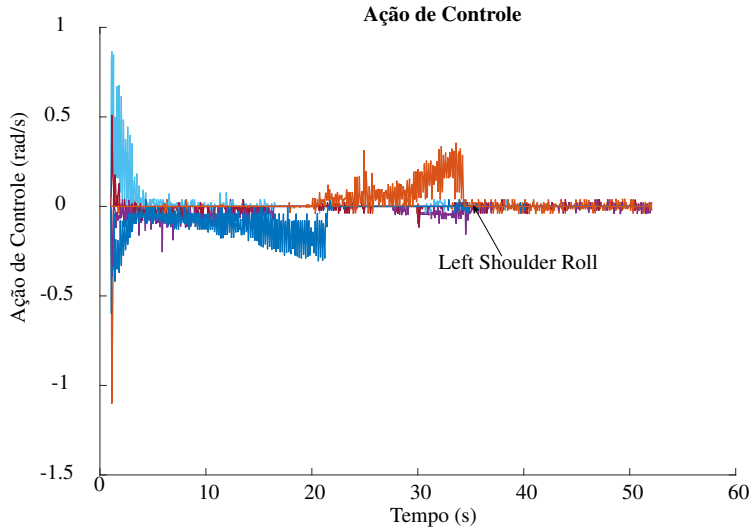
Pode-se confirmar que o robô chegou em um limite de junta ao se comparar o sinal de controle enviado pelo controlador e a ação executada pelas juntas, o que pode ser visto na Fig. 7.12.



(a) Vetor de juntas ao longo do tempo



(b) Sinal de controle enviado pelo controlador



(c) Ação de controle realizada pelas juntas

Fig. 7.12: Efeito do limite de juntas em uma tarefa no espaço cooperativo

Nota-se que apesar do controlador enviar um sinal, não houve resposta correspondente na ação das juntas. De fato, ao se analisar a Fig. 3.2b pode-se notar que a junta *Left Shoulder Roll* é limitada entre os ângulos -18^o e 76^o e verificando o gráfico do vetor de juntas (Fig. 7.12a) observa-se que esta junta está em -17^o. Sendo assim, não é mecanicamente possível mover essa junta além da posição em que ela se encontra.

7.6 CONTROLE DE DISTÂNCIA RELATIVA NO ESPAÇO COOPERATIVO

Dadas as diversas restrições do NAO, como o fato do robô possuir somente cinco graus de liberdade em cada braço — e portanto dez no espaço aumentado — um controle de posição fixa, em que a distância entre um braço em relação ao outro gastava três graus de liberdade, se mostrava inviável. Caso fosse decidido utilizar este tipo de controle outras tarefas não poderiam ser realizadas, assim optou-se por um controlador da norma da distância entre os braços em que utilizasse somente um grau de liberdade.

Como especificado na Sec. 6.4.4, foi projetado o controlador de distância relativa. Nesta seção são apresentados os resultados tanto da simulação quanto do robô físico.

7.6.1 Simulação

Este controlador possuía somente uma variável de controle, logo, a Fig. 7.13 mostra apenas o erro de uma distância entre os braços. Aqui o erro se encontra na ordem de 10^{-5}, mostrando que o controlador rapidamente converge para a distância desejada.

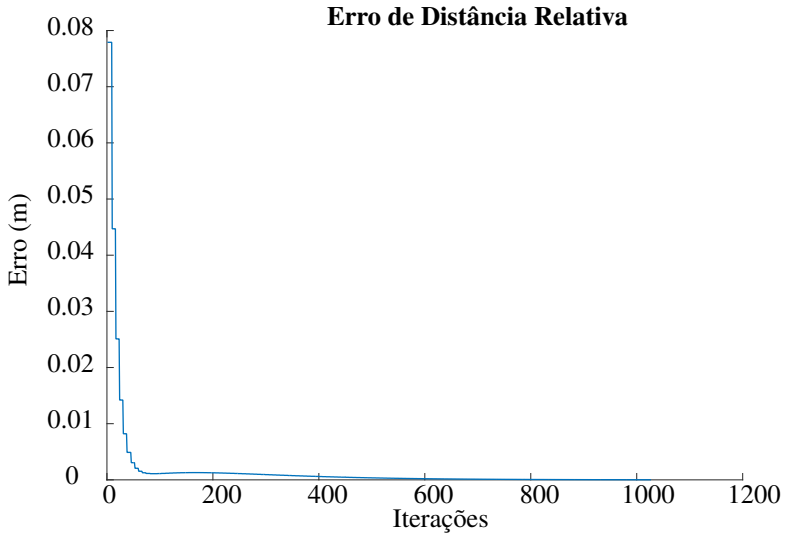
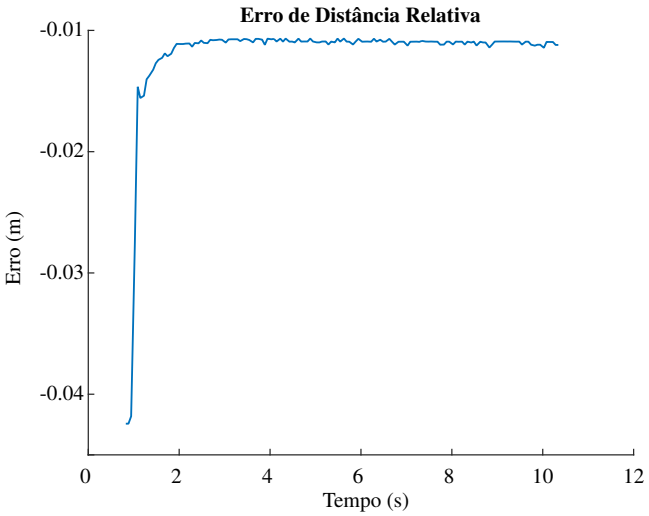


Fig. 7.13: Erro do controlador de distância relativa

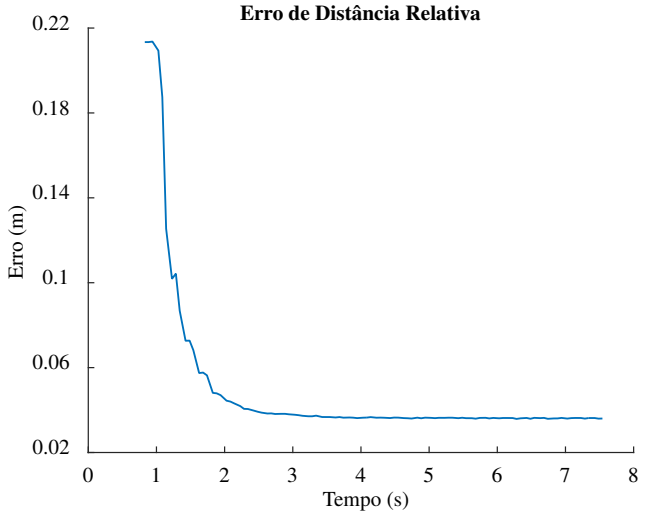
Nota-se que este controlador utiliza somente um grau e liberdade para a realização de uma tarefa no espaço cooperativo.

7.6.2 Plataforma Física

Na plataforma física foram testadas diferentes distâncias para o controle. A Fig. 7.14 mostra os erros com distâncias de referência maiores e menores que a distância entre os efetuadores na posição inicial (como na Fig. 3.3).



(a) Distância de 15cm



(b) Distância de 40cm

Fig. 7.14: Erro do controle de distância relativa

Nota-se que o erro ficou abaixo de 10% da distância desejada em ambas as tarefas.

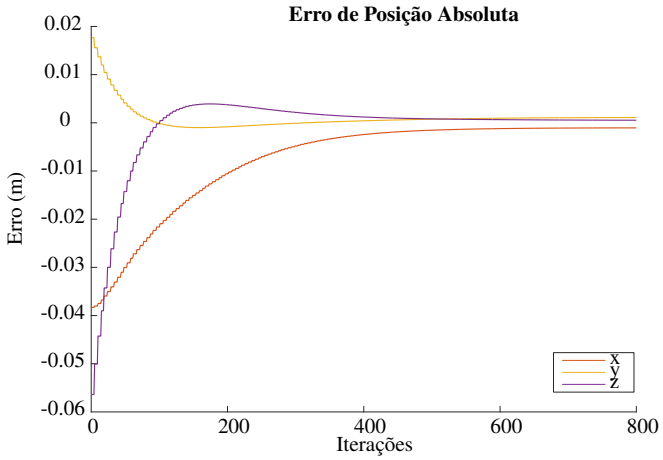
7.7 CONTROLE DE POSIÇÃO ABSOLUTA NO ESPAÇO COOPERATIVO

Como descrito na Sec. 6.4.5, foi implementado o controle de posição absoluta visando controlar o ponto médio entre os braços do robô, de modo que caso fosse necessária a movimentação de um objeto nas mãos do robô, sua posição poderia ser controlada diretamente.

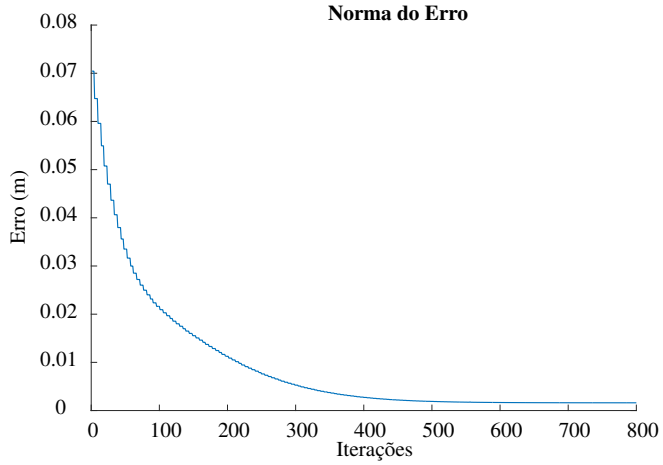
Testes foram feitos para o robô físico e simulado de modo que a posição de um ponto virtual entre seus manipuladores foi controlado.

7.7.1 Simulação

Aqui desejava-se controlar o ponto médio entre os efetuadores do robô, desse modo o erro utilizado para o cálculo do sinal de controle foi dado pela diferença entre um ponto no espaço e posição do ponto entre os efetuadores. A Fig. 7.15 mostra o erro e a sua norma.



(a) Erro da posição absoluta



(b) Norma do erro da posição absoluta

Fig. 7.15: Erro da Posição Absoluta

Verifica-se, a partir dos gráficos, que a simulação se estabilizou com um erro de aproximadamente 1.6mm.

É interessante também comparar os controladores implementados em simulação. Nota-se que o controle de orientação, posição relativa e distância relativa possuem um erro próximo de zero, portanto menor que os de posição para um único braço e de posição absoluta. Tal discrepância pode ser atribuída ao fato de que os controladores com menor erro usam somente o próprio robô

como referência, enquanto os outros utilizam alguma posição externa a qual nem sempre pode ser alcançada, assim resultando em erros maiores.

7.7.2 Plataforma Física

No caso do controle de posição absoluta implementado no NAO, pode-se observar os erros dados pela Fig. 7.17, em que a norma do erro se estabiliza em 1.2cm. Além disso a Fig. 7.16 mostra as posições de cada braço e a posição absoluta entre eles, de modo a facilitar a visualização deste ponto virtual.

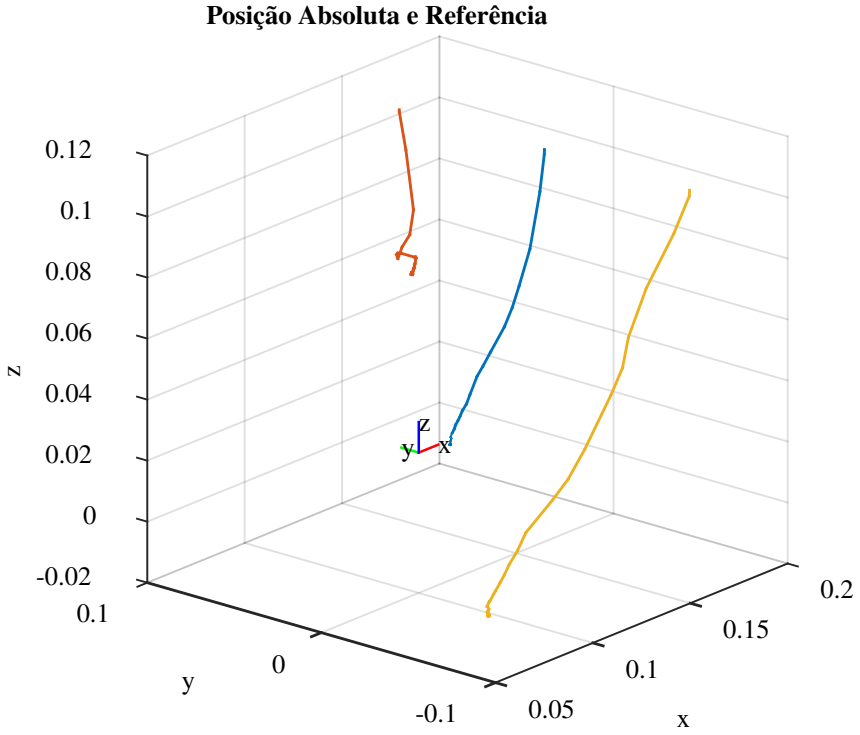
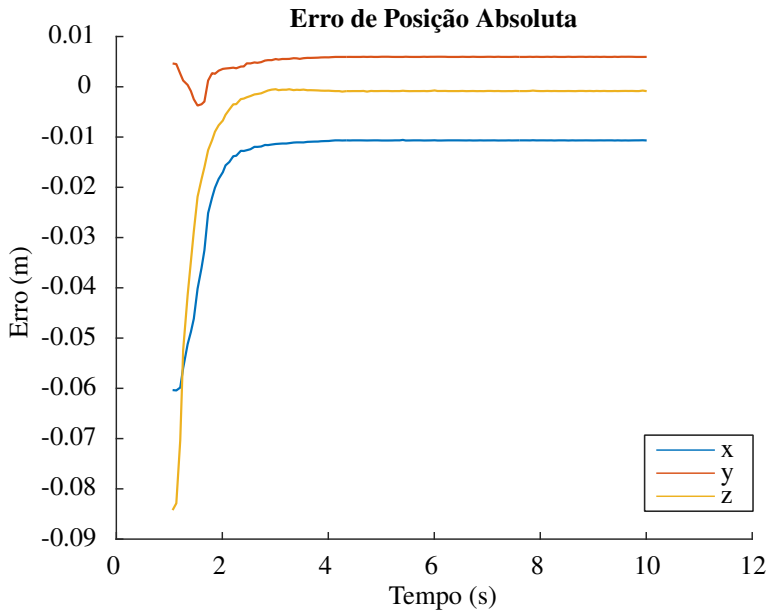
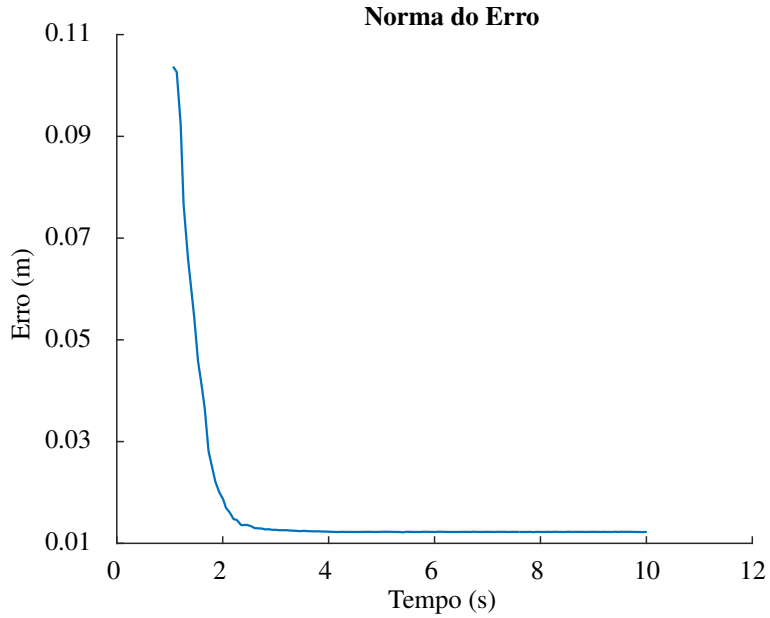


Fig. 7.16: Trajetória da posição absoluta entre os braços e posição de referência. Em azul é mostrada a posição absoluta, em vermelho a posição do efetuador esquerdo e em amarelo a posição do efetuador direito.



(a) Erro da posição absoluta



(b) Norma do erro da posição absoluta

Fig. 7.17: Erro do controlador de posição absoluta

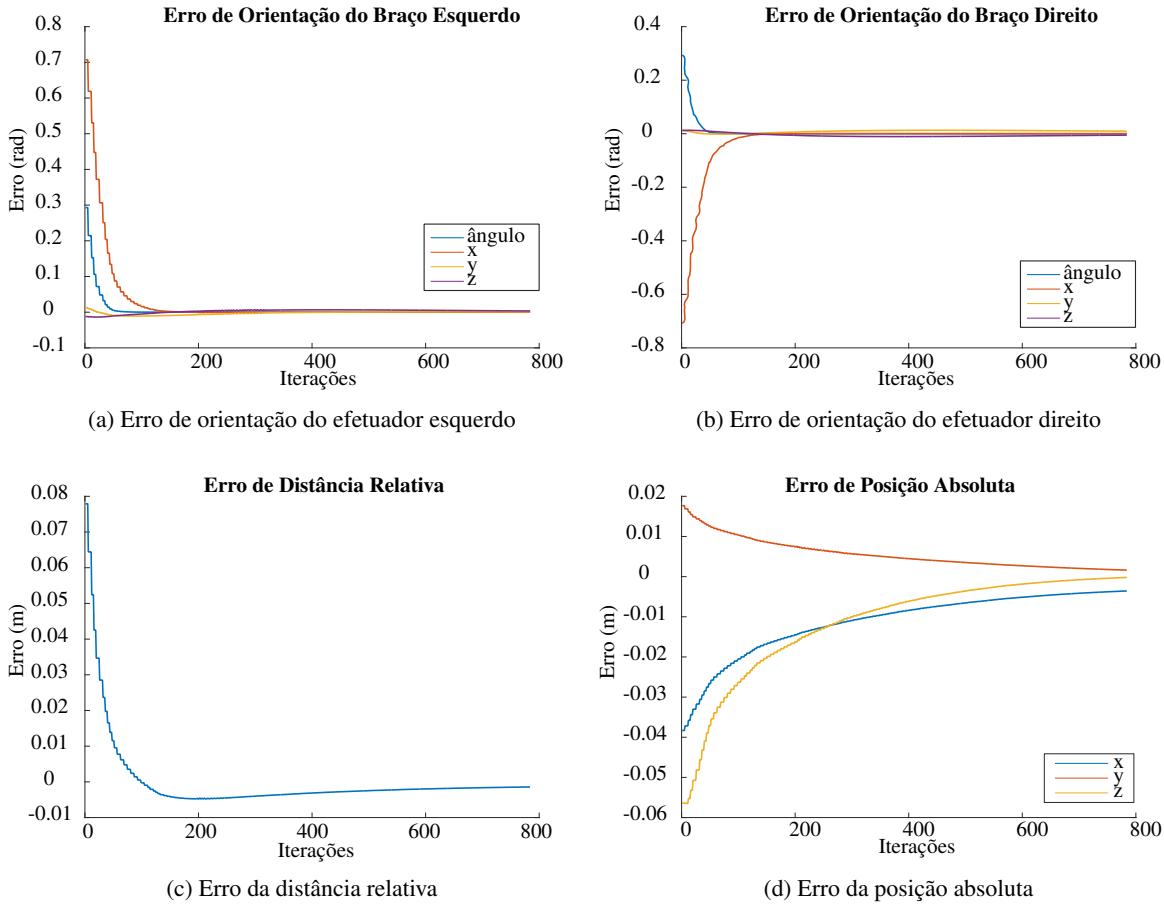
Observou-se então que a posição absoluta se estabilizava em um local diferente da referência, então utilizou-se a posição encontrada como referência de uma nova tarefa. O resultado obtido continuou se estabilizando em uma posição diferente da nova referência, o que pode ser explicado pelo fato do controlador implementado ser do tipo proporcional. Tais controladores se caracterizam por não resultar em erros nulos, mesmo que a posição desejada seja alcançável dentro do espaço de trabalho do robô.

7.8 CONCATENAÇÃO DE CONTROLADORES

Dados os diversos controladores implementados neste projeto, decidiu-se acoplá-los para a possível realização de uma tarefa, como descrito na Sec. 6.4.6. Nesta seção serão apresentados os resultados reais e simulados obtidos para tal controlador.

7.8.1 Simulação

Este controlador foi feito a partir da concatenação de todos os outros, a Fig. 7.18 mostra o erro de cada um dos controladores utilizados neste acoplamento: orientação de ambos os braços, distância relativa e posição absoluta exigindo todos os 10 DoFs disponíveis.



(a) Erro de orientação do efetuador esquerdo

(b) Erro de orientação do efetuador direito

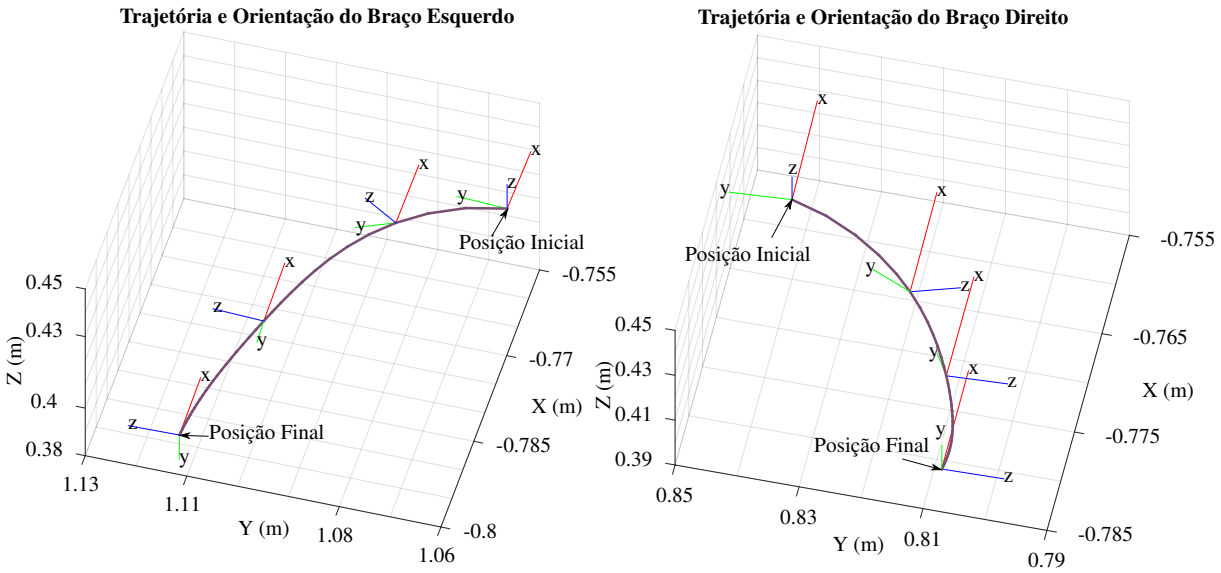
(c) Erro da distância relativa

(d) Erro da posição absoluta

Fig. 7.18: Erro de cada controlador utilizado no acoplamento

Neste controlador observa-se que há, de fato, uma convergência de todos os sinais, com erros se estabilizando na ordem de 10^{-3}. É interessante notar, neste caso que os erros são maiores que os erros dos controladores atuando sozinhos, o que é decorrente de um erro inerente do acoplamento. Em outras palavras, os controladores são concatenados em uma matriz de forma independente escalonados somente pelo erro, assim uma ação de controle pode interferir na outra causando erros maiores em uma das variáveis para que outra possa ser minimizada.

Além disso observa-se na Fig. 7.19 que a trajetória e a orientação de cada efetuador são controlados simultaneamente, o que mostra que todos os controladores de fato agem em conjunto.



(a) Braço Esquerdo

(b) Braço Direito

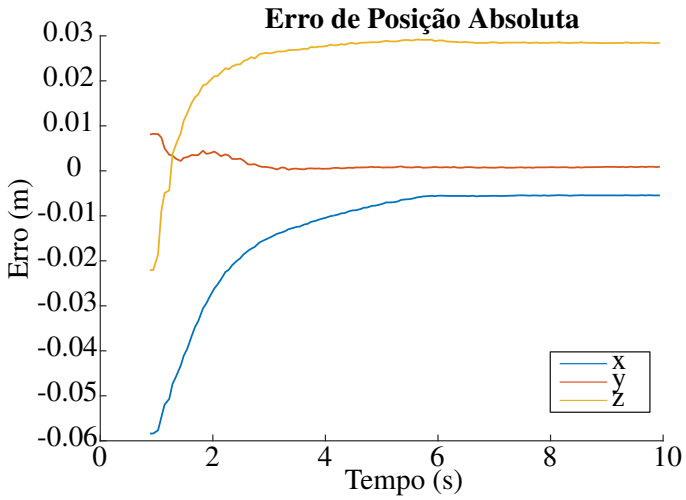
Fig. 7.19: Trajetória e Orientação do Efetuador

7.8.2 Plataforma Física

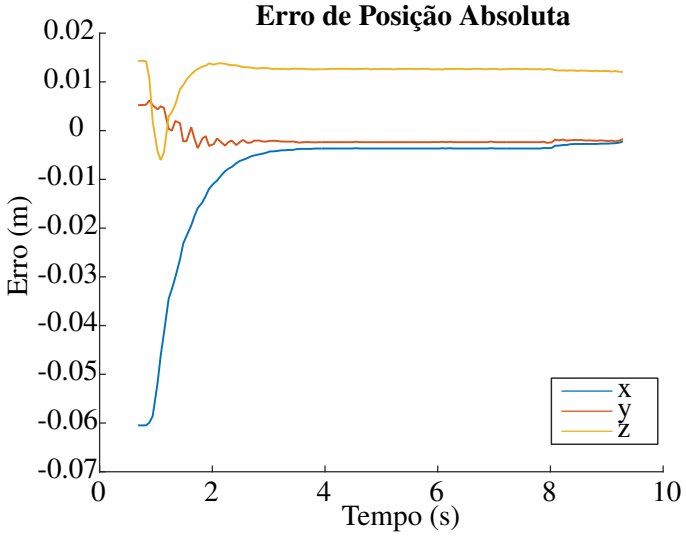
Para analisar o controlador concatenado, deve-se levar em conta o erro de cada variável separadamente. Desse modo, foram comparadas três tarefas diferentes, em que três posições e distâncias diferentes foram utilizadas como parâmetros de controle.

Primeiro, tem-se os erros da posição absoluta, dados pelos gráficos da Fig. 7.20 e suas normas, mostradas na Fig. 7.21. Nesta figura nota-se que a tarefa (c) possuiu o menor erro, chegando portanto mais próximo do seu objetivo, 0.1m na direção x. Para o controlador de distância relativa, mostrado na Fig. 7.22, o menor erro foi o da tarefa (a) que gerou 8mm de erro. Já controle de orientação, cujos erros são mostrados nas Figs. 7.23 e 7.24, se mostrou melhor no caso (b).

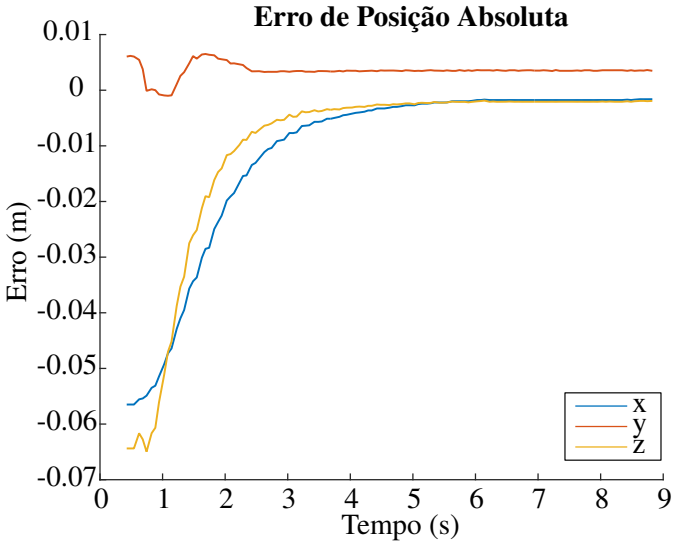
Aqui pode-se notar que muitas vezes, como dito na Sec. 7.8.1, para que uma das tarefas seja executada da melhor forma possível, outra poderá ficar com erros maiores em decorrência de ações de controle que causam interferências entre si. Observa-se, por exemplo, que enquanto a posição absoluta tem um erro baixo para a tarefa (c), a orientação do braço esquerdo para esta mesma tarefa possui um erro maior do que nos outros casos.



(a) Referência: $[0.1\ 0\ 0.05]\ m$

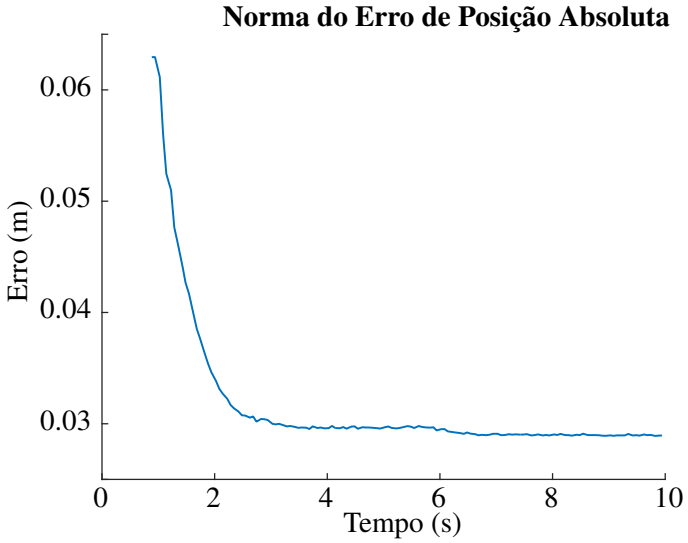


(b) Referência: $[0.1\ 0\ 0.1]\ m$

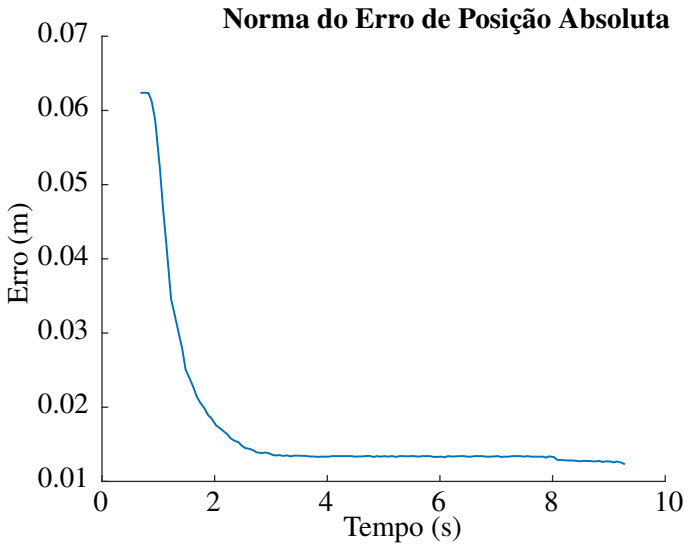


(c) Referência: $[0.1\ 0\ 0m]$

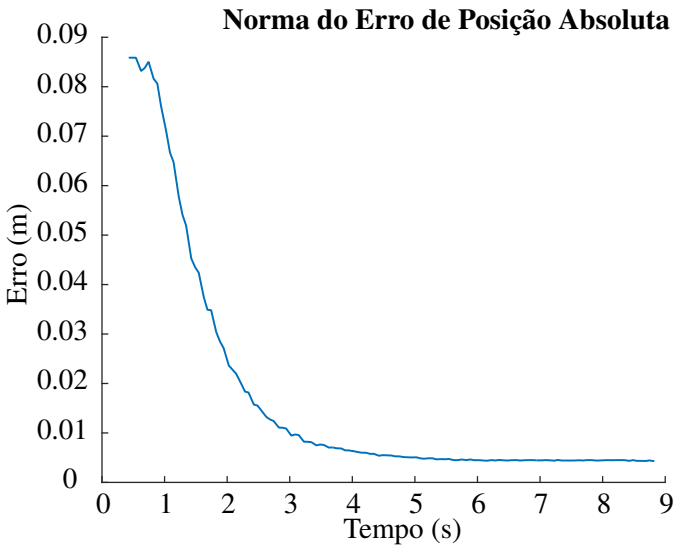
Fig. 7.20: Erro de posição absoluta



(a) Referência: $[0.1\ 0\ 0.05]\ m$

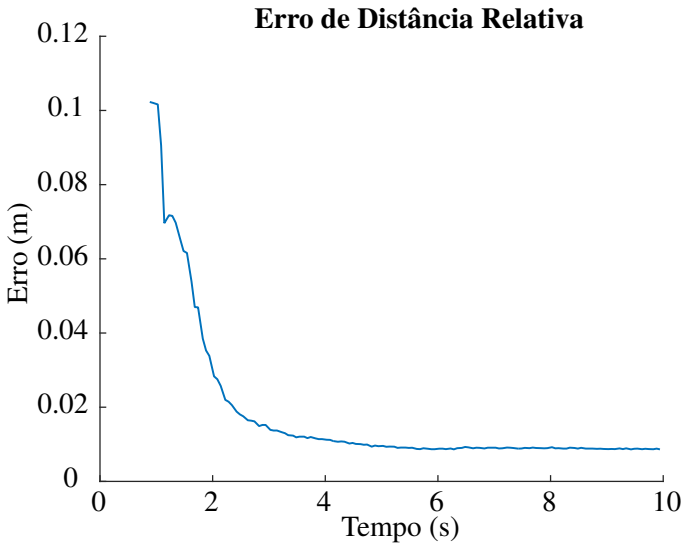


(b) Referência: $[0.1\ 0\ 0.1]\ m$

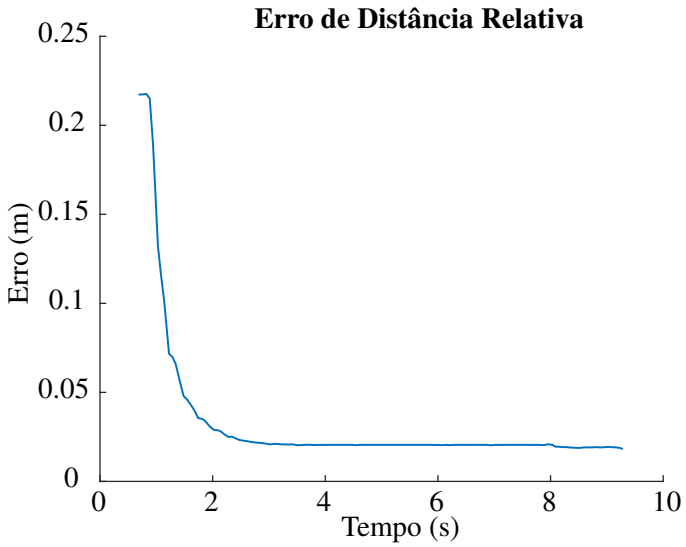


(c) Referência: $[0.1\ 0\ 0]\ m$

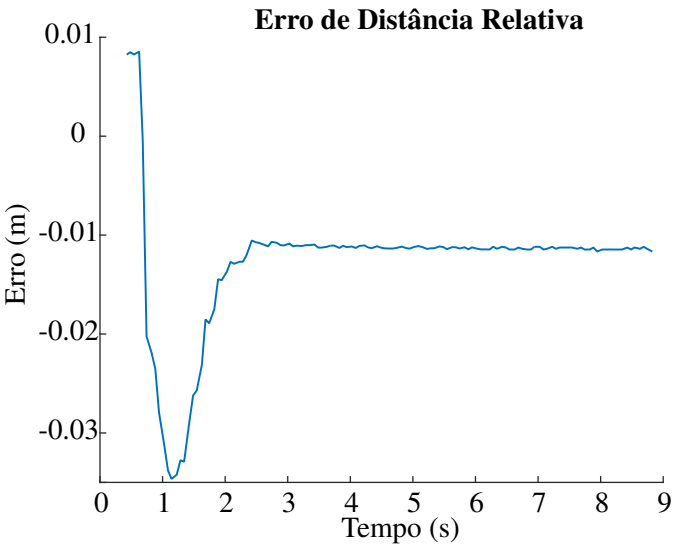
Fig. 7.21: Norma do erro da posição absoluta



(a) Referência: $0.3m$

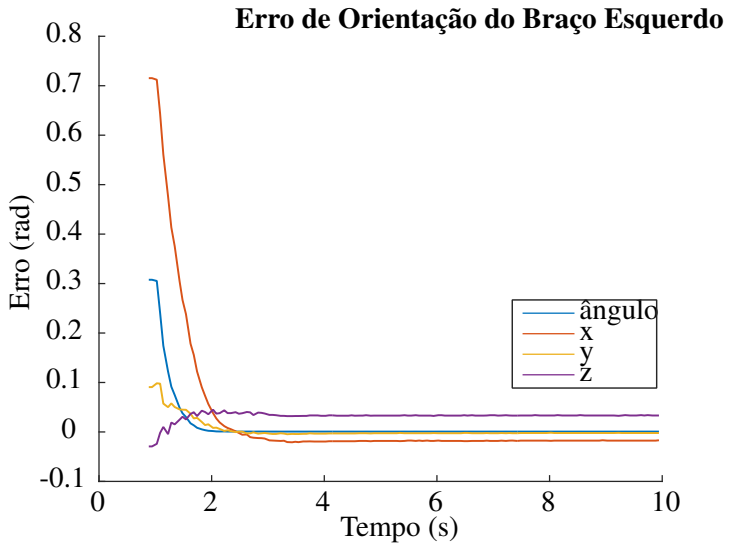


(b) Referência: $0.4m$

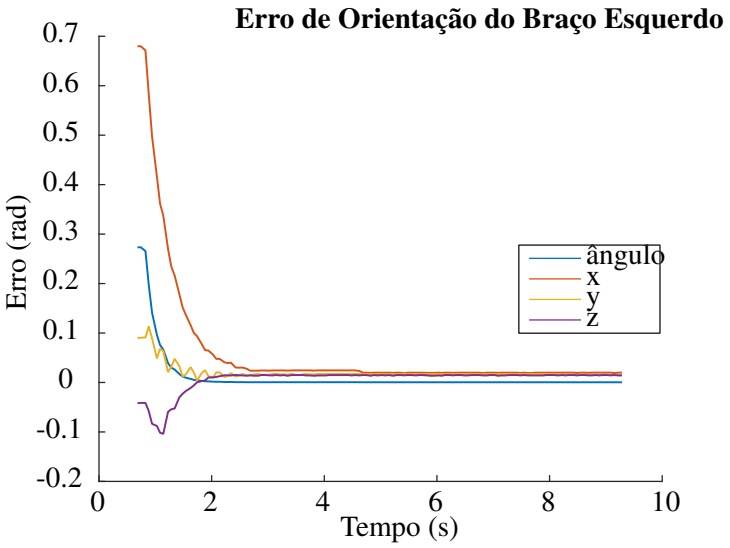


(c) Referência: $0.2m$

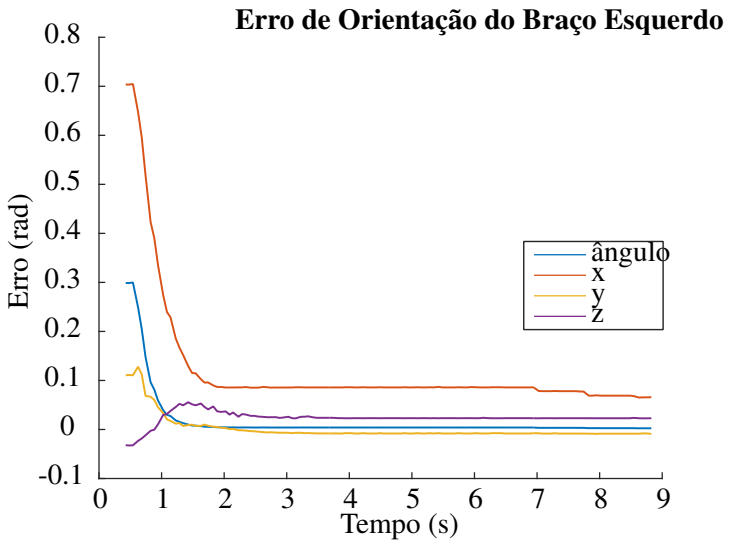
Fig. 7.22: Erro da distância relativa



(a)

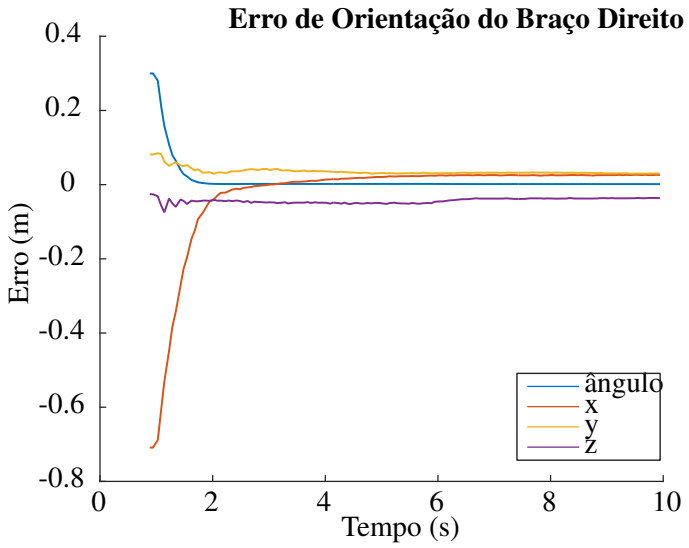


(b)

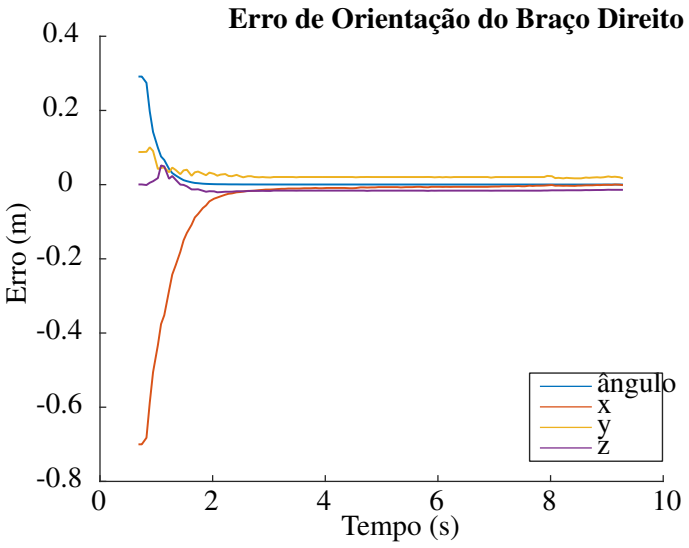


(c)

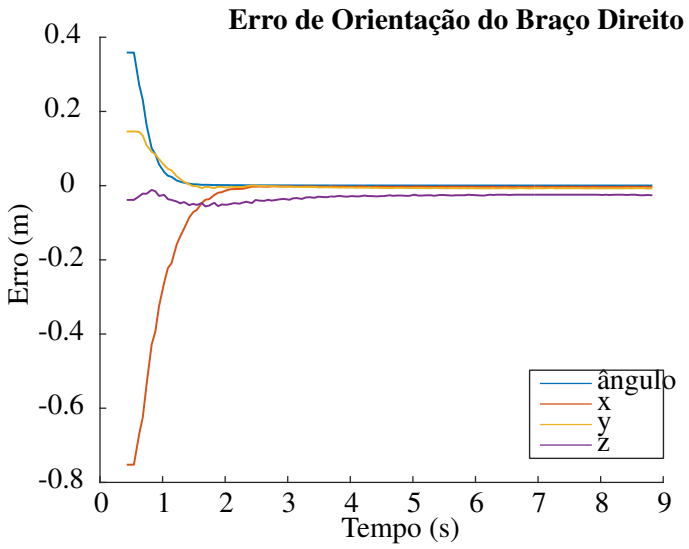
Fig. 7.23: Erro da orientação do efetuador esquerdo



(a)



(b)



(c)

Fig. 7.24: Erro da orientação do efetuador direito

Uma forma de se estabelecer melhor a hierarquia entre os controladores seria a partir da adição de controladores no espaço nulo do controlador da tarefa principal. Desta forma as ações de controle de tarefas designadas como secundárias não gerariam interferências na ação de controle da tarefa principal.

Uma análise interessante a ser feita, leva em conta o cálculo da pseudoinversa da Jacobiana. O pacote da DQRobotics possui uma função para este cálculo, entretanto, nesta implementação é calculada a pseudoinversa de Moore–Penrose definida em (2.89) e caso um ou mais de seus valores singulares se aproximem de zero o valor da pseudoinversa, que tenderia ao infinito, será truncado. Esse método causa descontinuidades na ação de controle, como pode ser visto na Fig. 7.26.

Um modo de se melhorar os resultados em posições singulares é a partir da introdução de um fator de amortecimento, como pode ser visto em (2.90). O fator de amortecimento, e consequentemente a Jacobiana amortecida, funcionam de forma a garantir a continuidade do sistema, mesmo quando há perda do posto matricial, o que ocorre próximo a regiões de singularidade[36]. A Fig. 7.25 mostra uma comparação entre as velocidades geradas por ambas as pseudoinversas.

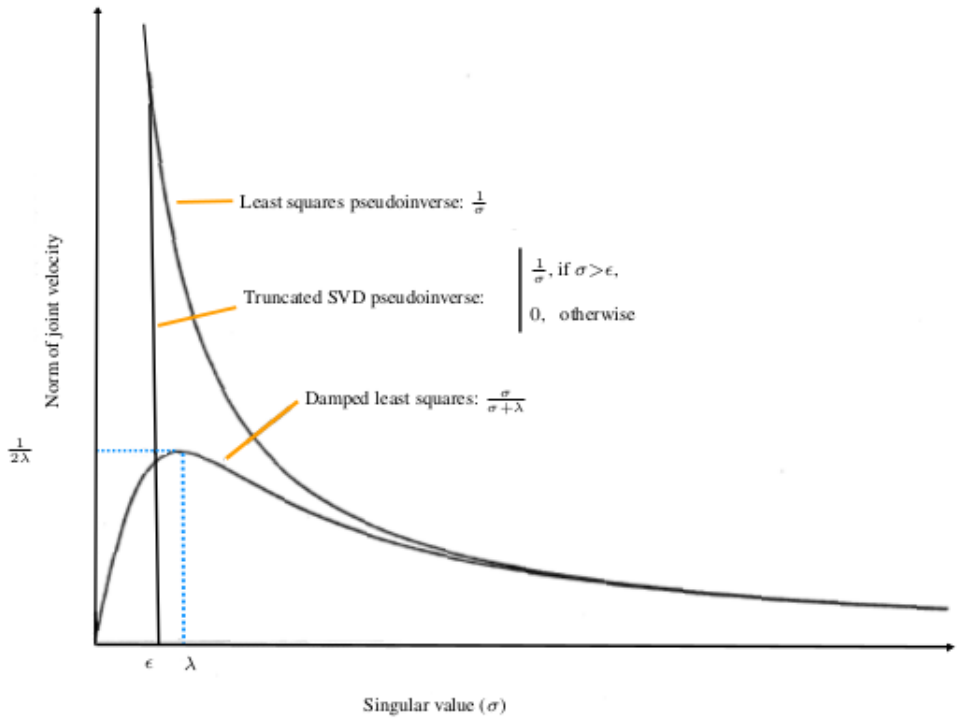
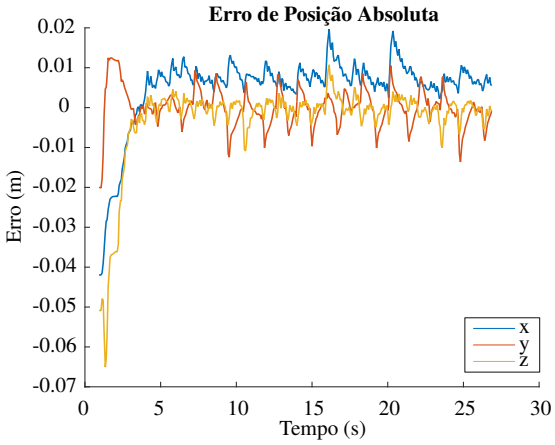
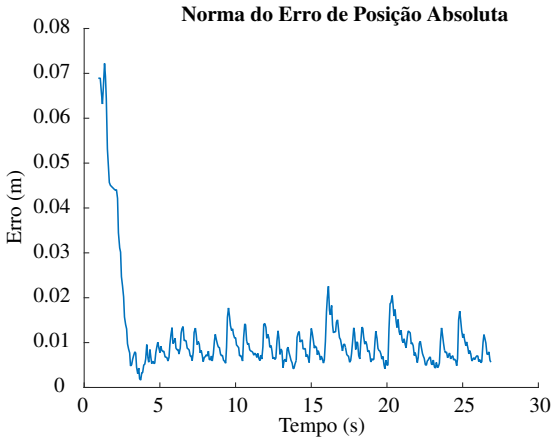


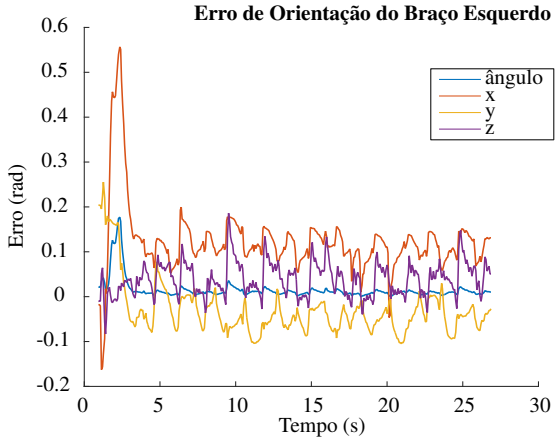
Fig. 7.25: Comparação entre o comportamento da pseudoinversa de Moore–Penrose, com truncamento em valores singulares, e da pseudoinversa amortecida. [6]



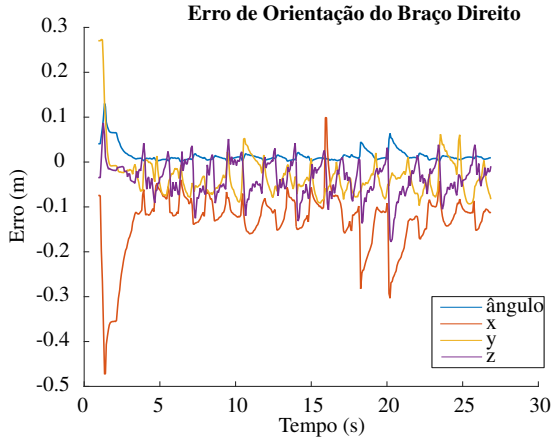
(a) Erro absoluto



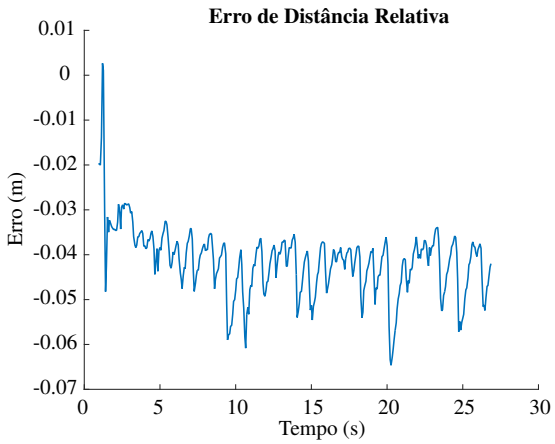
(b) Norma do erro absoluto



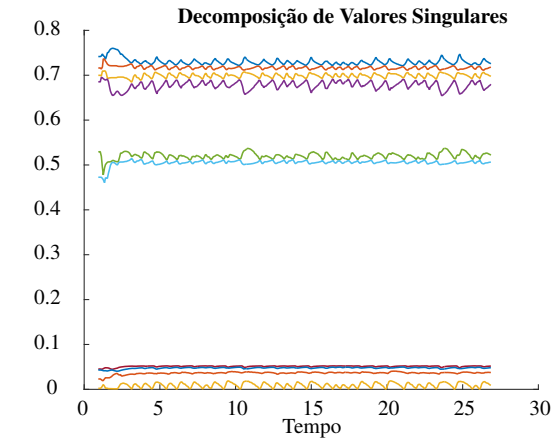
(c) Erro de orientação esquerdo



(d) Erro de orientação direito



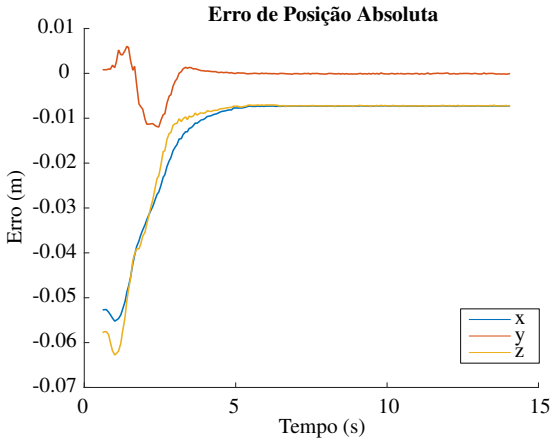
(e) Erro da distância relativa



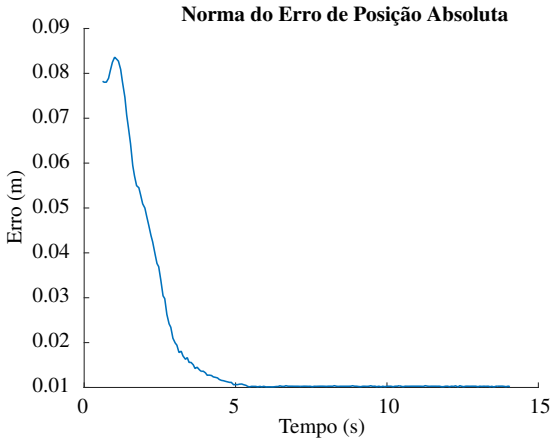
(f) Decomposição de valores singulares da Jacobiana

Fig. 7.26: Erro advindos da pseudoinversa truncada

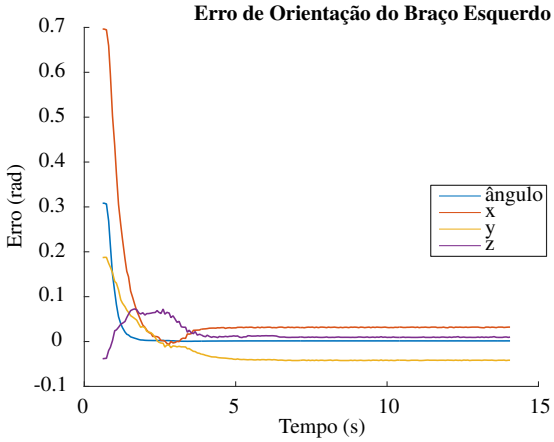
Devido as diversas restrições do NAO pode-se observar na Fig. 7.26f que nos momentos em que o robô entrava em uma região de singularidade sua ação de controle exibia descontinuidades levando a perturbações em seu movimento. De forma a verificar o comportamento da pseudoinversa amortecida realizou-se a mesma tarefa, partindo da mesma posição inicial, alterando apenas o cálculo da pseudoinversa. A Fig. 7.27 mostra a tarefa sendo realizada com a pseudoinversa amortecida.



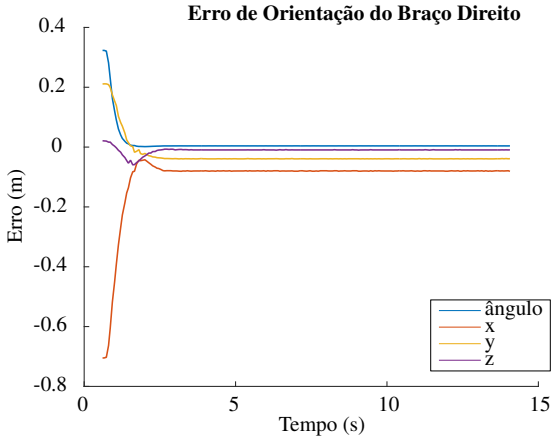
(a) Erro absoluto



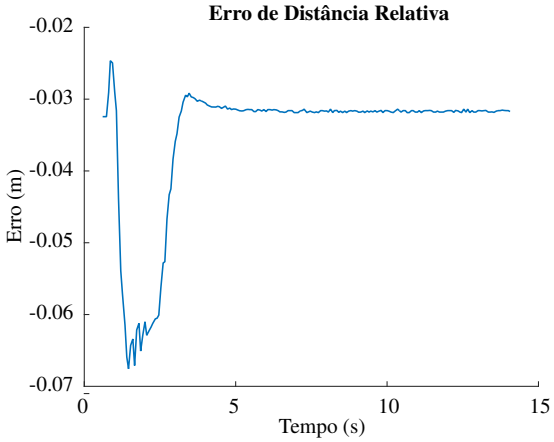
(b) Norma do erro absoluto



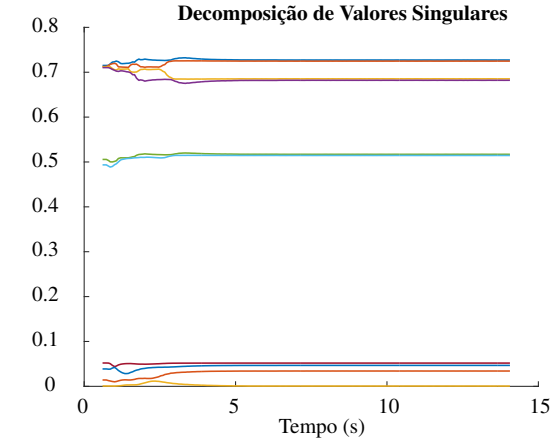
(c) Erro de orientação esquerdo



(d) Erro de orientação direito



(e) Erro da distância relativa



(f) Decomposição de valores singulares da Jacobiana

Fig. 7.27: Erro advindos da pseudoinversa amortecida

Outro problema encontrado ao se utilizar a pseudoinversa de Moore–Penrose truncada ocorre ao se estabelecer uma posição inicial em singularidade, como por exemplo a posição zero, mostrada na Fig. 6.1. Como esta pseudoinversa trunca as velocidades para zero, a ação de controle não consegue agir, resultando na ausência de movimento do robô.

8 CONCLUSÃO

"We can only see a short
distance ahead, but we can see plenty there that needs to be done."

Alan Turing

O presente trabalho teve como objetivo implementar tanto o ferramentas de comunicação visual quanto de controle no espaço cooperativo dual de modo a se criar aplicações em que robôs possam cooperar e se comunicar para a realização de tarefas, tanto em conjunto com outros robôs, com seres humanos ou realizando a cooperação entre seus dois manipuladores, como proposto na Sec. 1.2.

Para a comunicação utilizou-se técnicas de treinamento para detecção de objetos, levando em conta a detecção do robô como um todo e buscando encontrar melhoras nos resultados dividindo suas partes na imagem. Nessa etapa, foi constatado que apesar de alguns treinamentos serem capazes de detectar os objetos a curtas distâncias, não foi possível obter resultados satisfatórios para distâncias acima de 2m, devido a baixa resolução da câmera. Mesmo com resultados insatisfatórios em relação a detecção a longas distâncias, decidiu-se por criar e implementar um método de comunicação visual. Baseando-se na segmentação de LEDs do robô e utilizando um período de calibração antes de cada transmissão, foi possível enviar e receber com sucesso mensagens contendo posições cartesianas relativas a um campo de futebol.

Visando a cooperação entre robôs humanoides, o objetivo estabelecido foi a implementação de controladores utilizando quatérnios duais no espaço cooperativo para os dois manipuladores da plataforma NAO. O trabalho contou com um estudo das técnicas propostas por Adorno [2], no contexto de controle cooperativo e definição de primitivas de tarefas, e por Figueredo [6], no contexto de acoplamento de controladores e condicionamento numérico da Jacobiana. A validação do sistema foi feita a partir de simulações (utilizando Matlab e V-Rep) e dado o sucesso da implementação de cada controlador, estes eram testados no robô real, a partir de uma configuração feita com o sistema ROS.

Neste projeto, pode-se analisar a influência de diferentes ferramentas matemáticas aplicadas ao controle, como o espaço nulo e diferentes cálculos de pseudoinversas. Aqui foi possível comparar a pseudoinversa amortecida com a pseudoinversa de Moore–Penrose, de modo que enquanto a amortecida não introduz descontinuidades em posições singulares ela adicionava erros na trajetória final, que interferiam no comportamento do espaço nulo.

Os resultados obtidos para os controladores na plataforma física mostraram erros altos, indicando que, apesar da tarefa estar sendo executada, muitas vezes os efetuadores terminavam distantes de seus objetivos. Acredita-se que tais resultados podem ser atribuídos ao baixo número de DoFs do NAO e ao fato de seus motores possuírem folgas mecânicas consideráveis, acumulando erros na posição final do efetuador e criando uma zona morta[1] na ação de controle.

[1] Intervalo no domínio do sinal de controle que gera uma ação de controle nula

8.1 TRABALHOS FUTUROS

O campo da robótica é vasto e com diversas aplicações. Visando continuar pesquisas em métodos de comunicação em ambientes desprovidos de estruturas de rede e na cooperação entre robôs são propostas diversas expansões dos temas, como descritos nas seções a seguir.

8.1.1 Comunicação

- Desenvolver e implementar um protocolo para perda de mensagem;
- Introdução de redundâncias na mensagem como, por exemplo, um dígito verificador;
- Melhora na detecção do robô para distâncias superiores a 2m;
- Introdução de um sistema de mensagem e resposta. Onde ambos os robôs podem transmitir e receber mensagens.

8.1.2 Cooperação

- Utilizar a descrição de rotação relativa e absoluta para a orientação de um objeto, como definido em [2];
- Criar aplicação com robô manipulando objetos, permitindo, por exemplo, que ele carregue uma bandeja.
- Utilizar controladores mais sofisticados para compensar limitações mecânicas do robô. Por exemplo, pode ser introduzido um controle integral para compensar o baixo torque dos motores.
- Expandir modelo de controle, atualmente desenvolvido para apenas um robô, de modo a implementar tarefas cooperativas entre dois robôs humanoides (quatro manipuladores);
- Utilizar estrutura de mensagem visual para a comunicação entre dois robôs.

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[2] B. V. Adorno, "Two-arm manipulation: From manipulators to enhanced human-robot collaboration," Ph.D. dissertation, Université Montpellier II, Oct. 2011.

[3] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[4] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 7th ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[5] Aldebaran Robotics, 2015, [Acessado em 05/Fev/2016]. [Online]. Available: http://doc.aldebaran.com/

[6] L. F. C. Figueredo, "Kinematic control based on dual quaternion algebra and its application to robot manipulators," Ph.D. dissertation, Universidade de Brasília, July 2016.

[7] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, Mar. 2002. [Online]. Available: http://doi.acm.org/10.1145/504729.504754

[8] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[9] L. Perumal, "Quaternion and its application in rotation using sets of regions," *International Journal of Engineering and Technology Innovation*, vol. 1, no. 1, pp. 35–52, 2011.

[10] RoboCup Federation, "A brief history of robocup," 2016. [Online]. Available: http://www.robocup.org/

[11] A. K. Mackworth, "The dynamo project: The world's first robot soccer players," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2012, pp. 5442–5443.

[12] RoboCup Technical Committee, "Robocup standard platform league technical challenges," 2016. [Online]. Available: http://www.tzi.de/spl/pub/Website/Downloads/Challenges2016.pdf

[13] D. H. Ballard and C. M. Brown, *Computer Vision*, 1st ed. Prentice Hall Professional Technical Reference, 1982.

[14] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003.

[15] M. A. Treiber, *An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[16] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[17] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features." in *CVPR (1)*. IEEE Computer Society, 2001, pp. 511–518.

[18] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the Sixth International Conference on Computer Vision*, ser. ICCV '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 555–. [Online]. Available: http://dl.acm.org/citation.cfm?id=938978.939174

[19] P. J. Nahin, *An Imaginary Tale: The Story of i [the square root of minus one] (Princeton Library Science Edition)*. Princeton University Press, 1998. [Online]. Available: http://www.jstor.org/stable/j.ctt7sxsj

[20] T. Lam, "Hamilton's quaternions," *Handbook of Algebra*, no. 3, pp. 429–454, Dec. 2003.

[21] W. R. Hamilton, *Elements of Quaternions*. London: Longmans, Green, co, 1866. [Online]. Available: https://archive.org/details/elementsquaterni00hamirich

[22] L. Susanka, "An introduction to quaternions with application to rotations," Feb. 2014. [Online]. Available: http://susanka.org/Notes/quaternions.pdf

[23] A. A. Harkin and J. B. Harkin, "Geometry of generalized complex numbers," *Mathematics Magazine*, vol. 77, no. 2, pp. 118–129, Apr. 2004.

[24] J. Rooney, "William kingdon clifford (1845:1879)," in *Distinguished Figures in Mechanism and Machine Science: Their Contributions and Legacies*, M. Ceccarelli, Ed. Milton Keynes, UK: Springer Netherlands, 2004.

[25] R. Mukundan, "Quaternions: From classical mechanics to computer graphics, and beyond," in *Proceedings of the 7 th Asian Technology Conference in Mathematics*, 2002.

[26] B. Kenwright, "A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3d," in *The 20th International Conference on Computer Graphics, Visualization and Computer Vision, WSCG 2012 Communication Proceedings*, pp. 1–13.

[27] Y.-B. Jia, "Dual quaternion handout," Sep. 2015. [Online]. Available: http://web.cs.iastate.edu/~cs577/handouts/dual-quaternion.pdf

[28] R. M. Murray, Z. Li, S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Mar. 1994.

[29] M. P. M. Paulo Marcos de Aguiar, Glauco Augusto de Paula Caurin, "Análise cinestática de uma garra antropomórfica de três dedos utilizando a teoria helicoidal," in *II Congresso Nacional de Engenharia Mecânica*. ABMC, 2012.

[30] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley, *Computer graphics: principles and practice (3rd ed.)*. Boston, MA, USA: Addison-Wesley Professional, July 2013.

[31] S. Stramigioli and H. Bruyninckx, "Geometry and screw theory for robotics," *Tutorial during ICRA*, 2001.

[32] E. Lengyel, *Mathematics for 3D Game Programming and Computer Graphics, Third Edition*, 3rd ed. Boston, MA, United States: Course Technology Press, 2011.

[33] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices." *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955. [Online]. Available: http://ci.nii.ac.jp/naid/10008019314/en/

[34] J. J. Craig, *Introduction to robotics: mechanics and control*. Pearson Prentice Hall Upper Saddle River, 2005.

[35] N. Kofinas, "Forward and inverse kinematics for the nao humanoid robot," Master's thesis, Technical University of Crete, July 2012.

[36] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997. [Online]. Available: http://dx.doi.org/10.1109/70.585902

[37] L. F. C. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. A. Borges, "Switching strategy for flexible task execution using the cooperative dual task-space framework," in *2014 IEEE/RSJ Int. Conf. Intell. Robot. Syst.* Chicago: IEEE, 2014, pp. 1703–1709. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6942784

[38] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[39] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[40] The QT Company, "Qt documentation v. 4.02," 2016. [Online]. Available: https://www.qt.io/

[41] M. Jones and P. Viola, "Fast multi-view face detection," MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, Tech. Rep. TR2003-96, Aug. 2003. [Online]. Available: http://www.merl.com/publications/TR2003-96/

[42] T. DeWolf, "Robot control part 5: Controlling in the null space." [Online]. Available: https://studywolf.wordpress.com/2013/09/17/robot-control-5-controlling-in-the-null-space/