MURILO MENDONÇA VENÂNCIO

DESENVOLVIMENTO DE TÉCNICAS DE ESTABILIZAÇÃO DE CAMINHADA PARA ROBÔ HUMANOIDE COM DETECÇÃO DE DIFERENTES TIPOS DE TERRENOS

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA MECÂNICA

2018

MURILO MENDONÇA VENÂNCIO

DESENVOLVIMENTO DE TÉCNICAS DE ESTABILIZAÇÃO DE CAMINHADA PARA ROBÔ HUMANOIDE COM DETECÇÃO DE DIFERENTES TIPOS DE TERRENOS

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **MESTRE EM ENGENHARIA MECÂNICA.**

Área de concentração: Mecânica dos Sólidos e Vibrações.

Orientador: Prof. Dr. Rogério Sales Gonçalves.

UBERLÂNDIA - MG

2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

V448d 2018

Venâncio, Murilo Mendonça, 1992
Desenvolvimento de técnicas de estabilização de caminhada para robô humanoide com detecção de diferentes tipos de terrenos [recurso eletrônico] / Murilo Mendonça Venâncio. - 2018.

Orientador: Rogério Sales Gonçalves.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2018.857
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia mecânica. 2. Robótica evolutiva. 3. Redes neurais (Computação). 4. Robôs - Movimento. I. Gonçalves, Rogério Sales, , (Orient.) II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. III. Título.

CDU: 621

Angela Aparecida Vicentini Tzi Tziboy – CRB-6/947

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Engenharia Mecânica
Av. João Naves de Ávila, nº 2121, Bloco 1M, Sala 212 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4282 - www.posgrad.mecanica.ufu.br - secposmec@mecanica.ufu.br

ATA

ATA DE DEFESA DE DISSERTAÇÃO

NÚMERO DE ORDEM: 531

DATA: 28/09/2018

Às quatorze horas do dia vinte e oito de setembro de dois mil e dezoito, no Anfiteatro H do Bloco 5O, Campus Santa Mônica, reuniu-se a Banca Examinadora composta pelos Professores Dr. Rogério Sales Gonçalves (orientador) e Dr. Fran Sérgio Lobato da Universidade Federal de Uberlândia e, Dr. Reinaldo Augusto da Costa Bianchi do Centro Universitário FEI para, sob a presidência do primeiro, desenvolver o processo de avaliação da dissertação intitulada **"Desenvolvimento de Técnicas de Estabilização de Caminhada para Robô Humanoide com Detecção de Diferentes Tipos de Terreno"**, apresentada pelo aluno **MURILO MENDONÇA VENÂNCIO**, matrícula número **11622EMC015**, em complementação aos requisitos determinados pelo Regimento do Programa de Pós-Graduação em Engenharia Mecânica para obtenção do título de Mestre. Após discorrer sobre seu trabalho, o candidato foi arguido pelos membros da Banca, diante das comunidades universitária e externa. Em seguida, a dissertação foi avaliada em seção privada pelos membros da Banca que, ao encerrar o processo, consideraram-na:

( ) Aprovada

(X) Aprovada com modificações a serem submetidas para a aprovação do orientador.

( ) Aprovada com modificações a serem submetidas para a aprovação da banca.

( ) Reprovada

conferindo ao aluno, em caso de aprovação, o título de Mestre em Engenharia Mecânica, **Área de Concentração: Mecânica dos Sólidos e Vibrações, Linha de Pesquisa: Projetos de Sistemas Mecânicos**. As demandas complementares observadas pelos examinadores deverão ser satisfeitas no prazo máximo de 30 dias, para dar validade a esta aprovação. Para constar, lavrou-se a presente ata, que vai assinada pelo presidente e demais membros da Banca.

Membros:

Prof. Dr. Rogério Sales Gonçalves (orientador) - UFU

Prof. Dr. Fran Sérgio Lobato - UFU

Prof. Dr. Reinaldo Augusto da Costa Bianchi - FEI

Uberlândia, 28 de setembro de 2018

Documento assinado eletronicamente por **Rogerio Sales Gonçalves**, **Professor(a) do Magistério Superior**, em 28/09/2018, às 16:43, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Fran Sergio Lobato**, **Professor(a) do Magistério Superior**, em 28/09/2018, às 16:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Reinaldo Augusto da Costa Bianchi**, **Usuário Externo**, em 28/09/2018, às 16:50, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0732337** e o código CRC **03EAE678**.

Referência: Processo nº 23117.064291/2018-90 SEI nº 0732337

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado à vida e uma família maravilhosa que sempre apoiou minhas decisões

A Universidade Federal de Uberlândia pelo excelente curso de Engenharia Mecatrônica e de Pós-Graduação em Engenharia Mecânica, que me forneceu suporte e conhecimento ao longo dos anos de graduação e pós-graduação.

A Equipe de Desenvolvimento em Robótica Móvel (EDROM) por ter contribuído muito além deste projeto, por ter me motivado em dias difíceis, por todas as amizades que fiz, e pelos momentos alegres que ganhei durante os anos que fui membro da equipe.

Ao meu professor orientador Dr. Rogério Sales Gonçalves que sempre demonstrou confiar na minha capacidade, e mostrou que eu posso sempre superar barreiras mesmo que elas pareçam impossíveis. Além disso, o seu incentivo, sugestões, correções e cobrança pelos resultados contribuíram para que este trabalho fosse realizado.

A CAPES, inicialmente pela oportunidade de ter sido bolsista do programa Ciências sem Fronteiras, e estudar nos Estados Unidos, lugar onde pude ganhar uma visão mais abrangente a respeito da área da robótica, além de poder conhecer pessoas brilhantes que me inspiraram. E durante o meu mestrado, em que fui contemplado pela bolsa de estudos que me permitiu concentrar meus esforços durante a elaboração deste trabalho.

Gostaria de agradecer à minha namorada Ana Flávia, que teve paciência comigo, me deu sugestões, e me ajudou a ter ânimo em diversos momentos durante esses anos de mestrado.

Por fim, e com imensa importância, gostaria de agradecer à minha família, meus pais, Josivaldo e Laci, e minhas irmãs Josiane e Joice por terem me motivado em tempos difíceis, por serem minha inspiração, e por todo suporte essencial para as minhas realizações.

VENÂNCIO, M. M. **Desenvolvimento de Técnicas de Estabilização de Caminhada para Robô Humanoide com Detecção de Diferentes Tipos de Terrenos.** 2018. 132 f., Dissertação de Mestrado, Universidade Federal de Uberlândia, Brasil.

Resumo

A robótica humanoide tem sido impulsionada pelas possibilidades de utilização de robôs bípedes para auxiliar o ser humano em tarefas domésticas, de entretenimento e em ambientes hostis para o ser humano, como locais de desastres. Para conseguir lidar com os mais variados tipos de ambientes, é de grande importância que os robôs consigam desenvolver estratégias de locomoção que sejam adaptáveis com o meio em que está inserido. É neste contexto que este trabalho apresenta técnicas de locomoção humanoide, com a aplicação da estratégia de estabilização da locomoção humanoide e implementa uma abordagem nova na área de detecção de terrenos durante a caminhada humanoide. Para a geração das trajetórias de caminhada foi utilizado o método do pêndulo invertido linear em que o robô tem sua dinâmica simplificada como a de um pêndulo invertido. Este método demonstrou capacidade de geração de trajetórias estáveis para o modelo simulado do robô humanoide, já para o robô real, foram necessárias algumas adaptações do método para o ajuste da postura do robô e do período do tempo de cada passo. Em seguida, foi feito um estudo a respeito do controle de impedância, onde verificou-se que a modificação da constante de rigidez do controle PID (Proporcional Integral Derivativo) à nível de articulações tem a capacidade de melhorar a qualidade da caminhada do robô mesmo em diferentes tipos de terrenos. Por fim, foi proposta uma nova técnica de detecção de terreno, onde uma assinatura em formato de imagem monocromática do terreno é gerada por meio das informações dos sensores de torque e do sensor inercial presentes no robô humanoide. Através dessa assinatura, uma rede neural convolucional é utilizada para a classificação de seis diferentes tipos de terrenos, e a acurácia geral atingida foi superior à 96.0 %, que é comparável a técnicas presentes no estado da arte.

Palavras chave: Robôs humanoides, controle de impedância, método do pêndulo invertido linear, rede neural convolucional, classificação de terrenos.

VENÂNCIO, M. M. **Development of Stabilization Techniques for a Humanoid Robot with Detection of Different Types of Terrains.** 2018. 132 f., M. Sc. Dissertation, Universidade Federal de Uberlândia, Uberlândia.

Abstract

Humanoid robotics has been motivated by possibilities of using bipedal robots to assist humans in domestic tasks, entertainment and in hostile environments to humans, such as natural or human-induced disasters. To deal with the most varied types of environments, it is very important that the robots can develop adaptable locomotion strategies according to its surroundings. In this context, this work presents humanoid locomotion techniques, introducing a stabilization strategy for bipedal locomotion and implementing a new approach for terrain classification during walking. For the walking trajectories generation, the linear inverted pendulum method was used, and this method describes the robot's dynamics as a simple inverted pendulum. This approach demonstrated the ability of providing stable trajectories for a simulated model of the humanoid robot. On the other hand, some adaptations in the method was made for the real robot, such as adjusting the robot posture and changing the period of each step. Afterwards, a study was made on the impedance control, where it was noticed that changing the stiffness constant of the PID control at joint level has the ability to improve the walking stability of the robot, even in different types of terrains. Finally, a new terrain detection technique was proposed, where a gray scale picture is used as a terrain signature, and this image is generated by reading the data of torque sensors and the inertial sensor at the impact moment. A convolutional neural network is used to classify six different types of terrain using the signature picture as an input, and the overall accuracy of this method reached values above 96.0%, which is comparable to state of art approaches.

Keywords: Humanoid robots, impedance control, linear inverted pendulum method, convolutional neural networks, terrain classification.

LISTA DE SÍMBOLOS

α_i : parâmetros livres para o modelo do motor.

B : Constante de amortecimento.

$\boldsymbol{CM}$: Vetor da posição tridimensional do centro de massa.

δ : Parâmetro adimensional para superfície do material de atrito.

$e^{(k)}$: Erro na k-ésima iteração.

F : Força.

f_x : Força no eixo X.

f_y : Força no eixo Y.

f_z : Força no eixo Z.

$\boldsymbol{f}$: Vetor de força no espaço 3D.

g : Aceleração da gravidade.

γ : Fator de aceleração ou desaceleração da caminhada.

H_{step}: Altura do passo.

K : Constante de rigidez.

K_d : Constante de amortecimento do controlador PID.

K_p : Constante de rigidez do controlador PID.

M : Massa total do robô.

λ : Coeficiente de aprendizagem.

ω_0 : Velocidade angular desejada, ou de referência.

p_x, p_y : Posições do ZMP no plano XY.

$\boldsymbol{P}$: Posição e orientação para um dos pés do robô.

$\boldsymbol{P_{dir}}$**:** Vetor da Posição tridimensional do pé direito.

$\boldsymbol{P_{esq}}$**:** Vetor da Posição tridimensional do pé esquerdo.

$\boldsymbol{Q}$: Vetor da posição tridimensional do quadril do robô.

q_d : Posição desejada.

q_f : Posição obtida pelo método de *feedforward*.

$\dot{q}^{(s)}$: Velocidade de transição.

ξ : Distância intermediária entre dois pontos num plano.

$\rho(\xi)$: Força de reação vertical para a posição ξ.

$\sigma(\xi)$: Força de reação horizontal para a posição ξ.

$S_x(n)$: Distância do n-ésimo passo dado no eixo X.

S_y : Distância do passo dado no eixo Y.

T_C : Constante de tempo relacionada ao passo do robô.

T_d : Tempo de duplo apoio.

T_s : Tempo de simples apoio.

θ_0 : Posição angular desejada, ou de referência.

θ_i : Posição angular da articulação i.

θ, ϕ, γ : Ângulos de Euler.

τ : Torque.

U : Valor escalar da curva para processo de treinamento.

U_{bat} : Tensão na bateria.

v_x: Velocidade do robô no eixo X.

x_0 : Posição desejada, ou de referência.

$\dot{x}_0$: Velocidade desejada, ou de referência.

$\boldsymbol{W}$: Vetor com parâmetros relativos à perna do robô.

z_c : Altura do centro de massa do robô.

SUMÁRIO

1 INTRODUÇÃO 1
2 LOCOMOÇÃO BÍPEDE 9
2.1 Características da locomoção bípede 9
2.1.1 Fases de apoio 9
2.1.2 Polígono de apoio 10
2.1.3 Tipos de Caminhada 11
2.1.4 Critério de Estabilidade do Ponto de Momento Zero (ZMP) 11
2.2 Estratégias de Locomoção Bípede 13
2.3 Estratégias de estabilização da caminhada 14
3 IMPLEMENTAÇÃO DO *SOFTWARE* E DA SIMULAÇÃO DO ROBÔ HUMANOIDE 16
3.1 Simulação 16
3.2 *Software* desenvolvido 18
4 MÉTODO DO PÊNDULO INVERTIDO LINEAR 23
4.1 Mudanças no LIPM 30
4.2 Mudanças na postura do robô 31
4.3 Resultados e discussões 35
4.3.1 Análise de Sensibilidade 39
5 CONTROLE DE IMPEDÂNCIA 43
5.1 Mudança no ganho proporcional durante a caminhada 58
6 CARACTERIZAÇÃO DO TERRENO 63
6.1 Revisão de trabalhos na área 65
6.1.1 Robôs quadrúpedes 65
6.1.2 Robôs Hexápodes 65
6.1.3 Robôs Humanoides 66
6.2 Motivação para o método utilizado 66
6.3 Geração da assinatura do terreno 73
6.4 Redes Neurais Convolucionais 75
6.5 CNN utilizada 80
6.6 Resultados e discussões 81
7 CONCLUSÕES 85
7.1 Trabalhos futuros 86

REFERÊNCIAS BIBLIOGRÁFICAS 88
APÊNDICE A 94
APÊNDICE B 107
B.1 Cinemática inversa 107
B.2 Posição do centro de massa 111
APÊNDICE C 113
C.1 Inicialização de variáveis que serão atualizadas ao longo do processo: 113
C.4 Laço de repetição 116
APÊNDICE D 117
APÊNDICE E 120

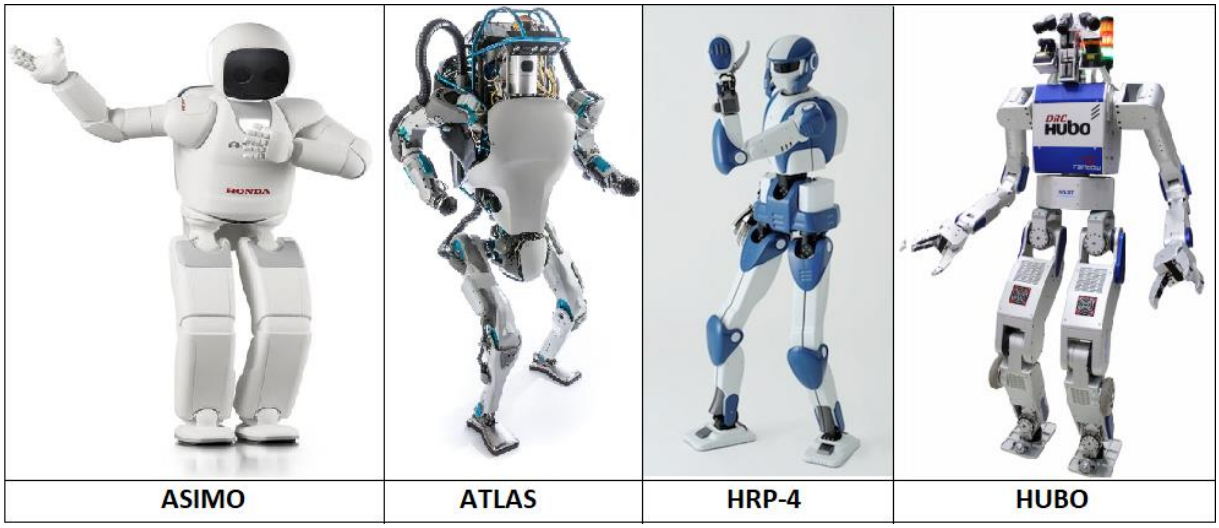
CAPÍTULO I

INTRODUÇÃO

O desenvolvimento de meios de locomoção mais eficazes, para os robôs móveis têm sido o foco de muitos estudos ao longo da história da humanidade. Um dos principais desafios é a mobilidade em ambientes criados para humanos, onde há escadas, corredores, calçadas, terrenos acidentados e demais obstáculos, em que a locomoção bípede tem tido sucesso em transpô-los. Devido à essa versatilidade, a pesquisa na área de robôs humanoides vem sendo explorada nas últimas décadas, sendo que as aplicações variam de desenvolvimento de exoesqueletos robóticos à robôs que podem auxiliar no resgate de vítimas de tragédias (WESTERVELT *et al.*, 2007).

A aparência do robô humanoide é semelhante ao corpo humano e, geralmente, apresentam tronco, cabeça, dois braços e duas pernas (MOUSSA, 2013). Além disso, são normalmente providos de sensores que auxiliam na percepção do mundo exterior, tais como: câmera, sensores inerciais, sensor de toque dentre outros.

A Figura 1.1 apresenta alguns robôs humanoides de grande importância nessa área. Um dos robôs de destaque é o ASIMO, que teve seu desenvolvimento iniciado na década de 80 pela empresa Honda. Atualmente, ele consegue manipular objetos, interagir com pessoas, e suas características motoras o permite correr e andar por terrenos irregulares (HONDANEWS, [S.d.]). O robô ATLAS, por sua vez, é desenvolvido pela empresa Boston Dynamics, e apresenta habilidades dinâmicas que o permite andar por terrenos acidentados, manter o equilíbrio mesmo em condições adversas do ambiente, e manipular objetos (DYNAMICS, 2018). O robô humanoide HRP-4 é uma plataforma para pesquisa e desenvolvimento de robôs humanoides desenvolvidos pela KAWADA Industries em colaboração com a AIST (*National Institute of Advanced Industrial Science and Technology*) (“Humanoid Robot HRP-4”, [S.d.]). Por fim, o robô Hubo, desenvolvido pela *Korea Advanced Institute of Science and Technology’s Humanoid Robot Research Center*, é capaz de caminhar sem manter os joelhos curvados, que apesar de ser dinamicamente mais estável, é menos natural do que a forma como os seres humanos caminham (GUIZZO, 2010).



(a) (b) (c) (d)

Figura 1.1 - Exemplos de robôs humanoides. (a) ASIMO (http://asimo.honda.com/gallery.aspx); (b) ATLAS (https://www.bostondynamics.com/atlas); (c) HRP-4 (http://global.kawada.jp/mechatronics/hrp4.html);(d) HUBO (https://www.robotshop.com/blog/en/drc-hubo-humanoid-robot-wins-the-darpa-robotics-challenge-16056), (Acessos em: 22/08/2018).

O desenvolvimento de robôs bípedes e avanços nas técnicas para uma locomoção estável tem motivado a criação de exoesqueletos robóticos. Estes mecanismos tem o potencial de auxiliar pessoas que possuem alguma deficiência motora, ou mesmo falta de algum dos membros do corpo. O projeto Andar de novo (do inglês *Walk again*), liderado pelo neurocientista Miguel Nicolelis (TONELLI; RESENDE, 2014), é um exemplo de aplicação de exoesqueleto robótico com o objetivo de reabilitar deficientes físicos, pacientes paraplégicos, para andarem por meio do exoesqueleto e de comandos oriundos do próprio cérebro do usuário, Figura 1.2.

Com objetivo de desenvolver a robótica no mundo, foi criado no ano de 1997 a *RoboCup*, uma competição internacional de robótica que contempla diversas categorias, dentre elas o futebol de robô humanoides ("*A Brief History of RoboCup*", [S.d.]). Nessa categoria, os robôs devem apresentar características de locomoção bípede, para lidarem com os diferentes desafios impostos, tais como: desenvolver uma caminhada estável e omnidirecional, chutar uma bola para marcar gols, localizar a bola e o gol no campo de futebol e desenvolver estratégias.

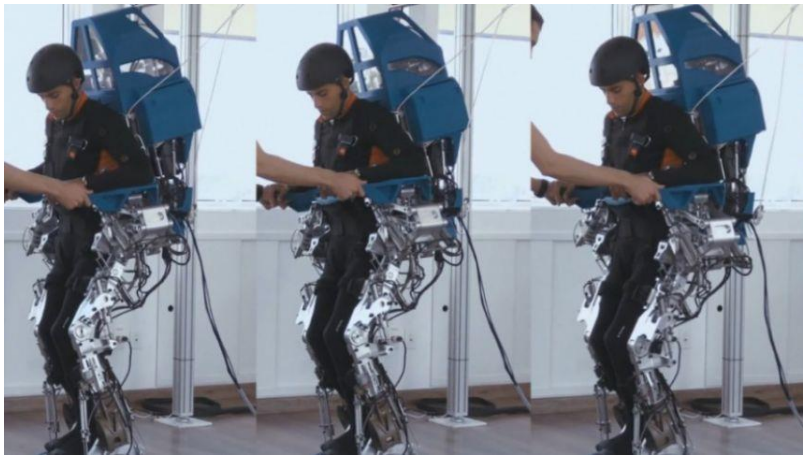


Figura 1.2 - O exoesqueleto Alberto Santos Dumont 1 do projeto Andar de novo (https://www.sciencesetavenir.fr/sante/cerveau-et-psy/les-exosquelettes-contribuent-a-la-recuperation-nerveuse_104457) (Acesso em 01/08/2018).

Ao longo dos anos a *RoboCup* tem aumentado o nível de dificuldade para as equipes participantes, desde o aumento do peso do robô, o tamanho da bola utilizada, e ultimamente, a mudança do material do campo, que passou de um carpete liso para grama sintética, o que desafia a capacidade dos robôs em manterem o equilíbrio durante a caminhada. O aumento gradual da dificuldade é um plano da *RoboCup* para que em 2050, a competição de futebol seja entre um time formado por robôs humanoides contra a melhor seleção de futebol de humanos da época (ROBOCUP, [S.d.]). Esta meta ambiciosa tem contribuído em muito para o desenvolvimento da robótica humanoide, uma vez que estimula equipes do mundo inteiro a investir neste tipo de pesquisa.

O objetivo desta dissertação é o desenvolvimento de uma caminhada bípede estável com capacidade de identificação de diferentes tipos de terrenos. Além disso, será discutido formas de adaptar alguns parâmetros da caminhada com o objetivo de estabilizar o movimento para diferentes tipos de terrenos, como por exemplo, a alteração da rigidez do controlador a nível de articulação.

Para a simulação da caminhada bípede e dos diferentes tipos de terrenos foi utilizado o *software* Gazebo (TAKAYA *et al.*, 2016), que permite a modelagem do robô através de um arquivo com a descrição das características dinâmicas do robô, e a configuração de sensores inerciais e de torque, com a leitura de seus valores durante a execução dos movimentos de caminhada. A utilização desse simulador teve papel importante nos testes das técnicas apresentadas ao longo dessa dissertação.

O método utilizado para a identificação do terreno teve como motivação a distinção visual das imagens mostradas na Figura 1.3, onde cada ponto da imagem representa um valor no tempo dos sensores utilizados na identificação. Os sensores utilizados são um

sensor IMU e dois sensores de torque por perna, sendo que as linhas horizontais na Figura 1.3 representam os valores lidos na seguinte ordem de cima para baixo: torque no joelho, torque no calcanhar, aceleração linear lateral do robô, velocidade angular e posição angular do ângulo que o robô faz lateralmente. No eixo horizontal da Figura 1.3, cada coluna representa os valores dos sensores no tempo ao longo do período de 0.244 s, que foi um intervalo determinado com base no tempo que o pé do robô permanece no chão a cada passo. Da Figura 1.3 percebe-se diferenças entre os terrenos que são evidentes e formam uma espécie de assinatura do terreno. Esse fato inspirou a utilização de Redes Neurais Convolucionais, que tem sido amplamente utilizada para classificação de imagens (CIREŞAN *et al.*, 2011). Essa abordagem demonstrou velocidade de processamento adequado para implementação online, com detecção de terreno com taxa de acerto em torno de 96% para seis terrenos, tanto em ambiente simulado, quanto em um robô humanoide real.

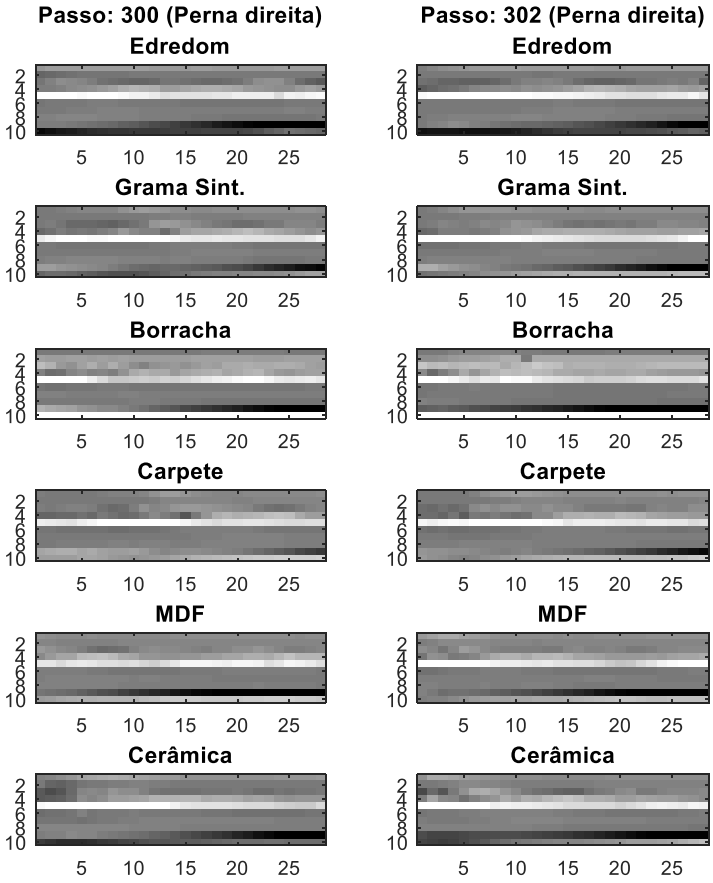


Figura 1.3 – Imagens construída com a informações de sensores após o impacto para terrenos de diferentes características. Estas imagens fornecem uma assinatura para cada ambiente.

Este é um trabalho de interesse da Equipe de Desenvolvimento em Robótica Móvel (EDROM) da UFU, e para as pesquisas desenvolvidas na área de biomecânica.

Para atingir os objetivos propostos nessa dissertação, a metodologia aplicada é capaz de gerar as trajetórias cartesianas que cada efetuador, no caso os pés, deverão seguir, e converter estas informações para coordenadas de articulações para cada servomotor do robô e então, durante a locomoção o robô deverá ser capaz de caracterizar a rigidez do solo em que se encontra através do processamento dos dados provenientes tanto do torque de alguns servomotores, quanto dos valores referentes à aceleração linear, velocidade de rotação e posição angular de um ponto central no quadril.

Devido à inerente instabilidade da caminhada humana, foi aplicado a redução do ganho proporcional no controle PID dos motores com o objetivo de melhorar a capacidade de rejeição de distúrbios durante a caminhada, melhorando a performance. Esse efeito se mostrou benéfico para a redução de picos de correntes e testes com diferentes valores de rigidez mostraram a melhoria da estabilidade para uma certa faixa de rigidez mesmo com a caminhada em diferentes tipos de terrenos simulados.

A locomoção de robôs humanoides em superfícies planas e sem perturbações já foi tópico de pesquisas que já demonstraram resultados no estado da arte da locomoção bípede (DALEN, 2012; KAJITA *et al.*, 2014). Robôs humanoides podem ter até mais de 20 graus de liberdade, o que dificulta inferir trajetórias estáveis por meio de seu modelo matemático completo, e isso motiva a busca e utilização de modelos que simplificam a estrutura antropomórfica do robô humanoide.

O modelo de Pêndulo Invertido Linear, ou simplesmente LIPM (*Linear Inverted Pendulum Method*), por exemplo, modela o comportamento dinâmico do robô como um pêndulo que tem sua base no pé de apoio durante a caminhada, e a massa localizada no centro de massa. Este modelo é capaz de gerar trajetórias para o centro de massa do robô a partir das coordenadas de onde os pés devem ser posicionados durante a locomoção. Através dessa relação de posições, é possível obter as trajetórias angulares para todas as articulações do robô através da cinemática inversa. A estabilidade dinâmica deste modelo é demonstrada matematicamente para o caso do pêndulo ideal, e sua aplicação em robôs humanoides reais tem demonstrado resultados promissores (DALEN, 2012; KAJITA *et al.*, 2014). Outros modelos utilizados para a simplificação da dinâmica bípede são abordados em (PRATT; DILWORTH; PRATT, 1997), (POULAKAKIS; GRIZZLE, 2009).

Uma abordagem comumente utilizada para a geração de movimento de caminhada bípede são as CPG (do inglês, *Central Pattern Generator*). Basicamente, essas estratégias baseiam-se na utilização de equações que descrevem padrões de movimentos que as articulações devem executar para se locomover (HONG; PARK; KIM, 2014). Essas

equações são geralmente de característica cíclica e são parametrizadas, possuindo algumas constantes que devem ser encontradas utilizando alguma metodologia empírica, ou mesmo de *Machine Learning* (MORADI; FATHIAN; SHIRY GHIDARY, 2016). É importante notar que estes métodos não são baseados diretamente na dinâmica do robô como no caso do LIPM, mas sim na simplificação do movimento de caminhada em partes mais simples.

Nesta dissertação, a abordagem utilizada para a geração de trajetórias será a utilização do modelo do pêndulo invertido linear que apresenta um conjunto intuitivo e reduzido de parâmetros a serem escolhidos quando comparado com métodos CPG. Além disso, esta abordagem é apontada como energeticamente eficiente, pois tem base na dinâmica natural do pêndulo invertido (DALEN, 2012). Este método também demonstrou ser promissor para a simulação do robô EVA (VENÂNCIO, 2016), que tem grande semelhança estrutural e de *hardware* em comparação com o robô utilizado neste trabalho.

A Figura 1.4(a) apresenta o esquema cinemático do robô utilizado nesta dissertação e a Figura 1.4(b) o protótipo do robô. Ele tem 89 cm de altura e pesa 7.2 kg. O robô possui 20 graus de liberdade no total, sendo seis em cada perna, três em cada braço e dois para os movimentos da cabeça. Os motores utilizados são do modelo MX-106 para os membros superiores e inferiores e do modelo MX-64 para o movimento da cabeça, sendo todos da mesma fabricante Dynamixel (ROBOTIS, 2018). Esses motores são do tipo servomotor digital, e permitem sua operação por meio de comunicação TTL com o controlador. O controlador utilizado é o Mini PC Intel® NUC que possui um processador Intel Core i5 de 1.6 GHz e 4GB de memória RAM. A estrutura mecânica do robô foi feita em maior parte de alumínio. O robô conta ainda com um IMU (unidade de medição inercial, do inglês *Inertial Measurement Unit*) localizado próximo ao seu centro de massa, que se trata de um sensor de inércia que fornece informações de velocidade angular e aceleração linear do ponto em que é instalado (STEPHANIE *et al.*, 2018). O projeto e a construção do robô foram executados pela EDROM.

Além do robô real, foi utilizado um modelo simulado para os testes de caminhada, da mudança no controle de impedância e para os testes de identificação do terreno. A utilização do modelo virtual do robô é importante para não correr riscos de danificar o *hardware* do robô real. Contudo, todas as técnicas aqui utilizadas, inclusive relativas aos *softwares* desenvolvidos, foram totalmente projetados para serem executados em ambas plataformas.

Algumas limitações mecânicas do robô real como: folgas mecânicas, torque insuficiente para alguns movimentos e algumas deficiências do próprio controle interno dos motores, fazem com que a proposta para o presente trabalho seja desafiante para a sua aplicação no mundo real utilizando os recursos disponíveis. Com o objetivo de contornar a

maior parte destes obstáculos, além da utilização da simulação para a maioria dos testes, foram introduzidas algumas alterações no método de geração de trajetórias de caminhada que são devidamente justificados no Capítulo IV, além disso, a adição de elementos elásticos em determinadas articulações demonstrou efeito promissor para a redução de folgas mecânicas.

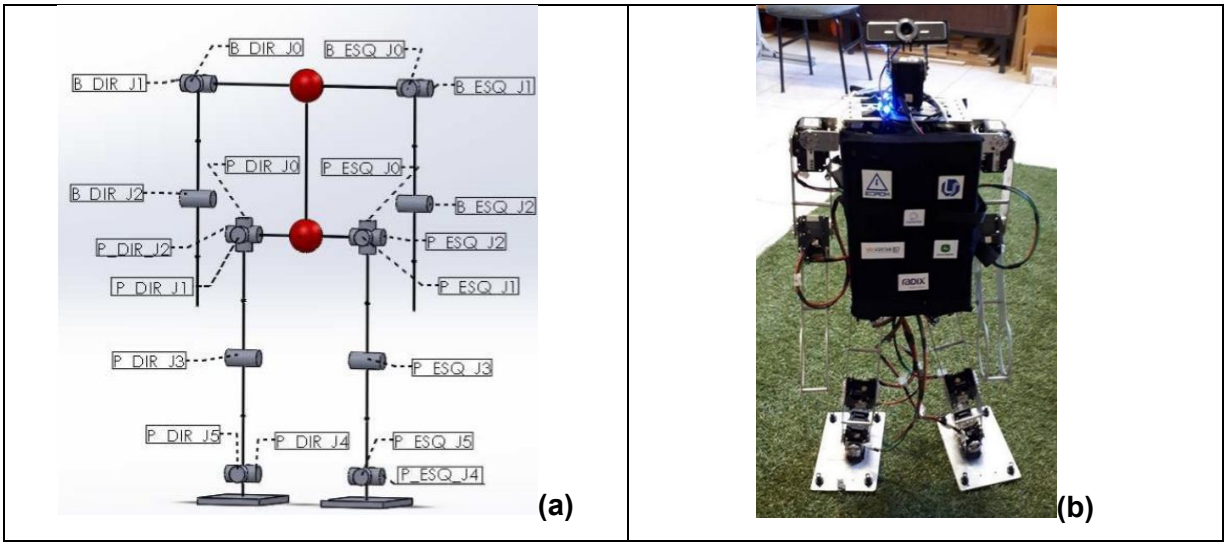


Figura 1.4 - (a) Esquema cinemático do robô a ser utilizado nessa dissertação com a nomenclatura das articulações sem considerar a estrutura da cabeça (VENÂNCIO, 2016); (b) Protótipo do robô.

Devido à pluralidade de assuntos abordados nessa dissertação, foi escolhido que cada capítulo tivesse sua própria revisão bibliográfica, testes e discussões. Os conteúdos foram abordados na seguinte ordem:

- Capítulo II: faz uma revisão a respeito dos principais conceitos relacionados à locomoção bípede, que são necessárias para o entendimento das nomenclaturas utilizadas ao longo do trabalho.

- Capítulo III: traz uma visão geral a respeito de como foi elaborado o *software*, além de algumas características do simulador utilizado e o modelo do robô humanoide.

- Capítulo IV: explica com maiores detalhes a teoria a respeito da aplicação do método LIPM, e trata de algumas modificações que foram feitas com o objetivo de deixar o método mais parametrizável para sua utilização no robô humanoide real. Os resultados da aplicação do método no robô humanoide real e no simulado são apresentados e discutidos.

- Capítulo V: é introduzido o conceito de controle de impedância e como ele foi empregado no robô. É mostrado alguns resultados da sua eficácia energética em testes práticos para um grau de liberdade, e em seguida é mostrado que a redução da rigidez no controle PID dos servomotores da perna do robô tem eficácia na estabilização da caminhada.

- Capítulo VI: é feita uma revisão a respeito dos métodos de detecção de terreno em diferentes estruturas de robôs móveis, e posteriormente é introduzido um novo método aplicado para a identificação do terreno, onde as redes neurais convolucionais são devidamente explicadas no contexto de classificação de imagens. Por fim são apresentados os resultados obtidos com as devidas discussões.

- Capítulo VII: serão apresentadas as conclusões deste trabalho e discutidas as possibilidades e sugestões de trabalho futuro.

CAPÍTULO II

LOCOMOÇÃO BÍPEDE

2.1 Características da locomoção bípede

Ao estudar a locomoção bípede aplicada para robôs humanoides, alguns termos aparecem com frequência, e entender seus conceitos é importante para a compreensão das abordagens empregadas tanto na geração de trajetórias quanto na estabilização do movimento. Os termos serão brevemente explicados neste Capítulo para auxiliar a compreensão dessa dissertação.

2.1.1 Fases de apoio

A caminhada humanoide é caracterizada pela constante troca dos pés de apoio, com momentos em que apenas um pé toca o chão, Figura 2.1(a), enquanto o outro se prepara para ser o próximo pé de apoio. Esse período é comumente referido como fase de simples apoio, ou SSP (*Single Support Phase*) (KAJITA *et al.*, 2014).

Além do SSP, a caminhada bípede apresenta a fase de duplo apoio, ou DSP (*Double Support Phase*) em que os dois pés estão em contato com o chão no mesmo instante, Figura 2.1(b). Geralmente isso ocorre no início da caminhada, e durante algum período após a aterrisagem do pé que estava no ar.

Ambas fases são importantes para a análise da estabilidade do robô em conjunto com o conceito de Polígono de Apoio.

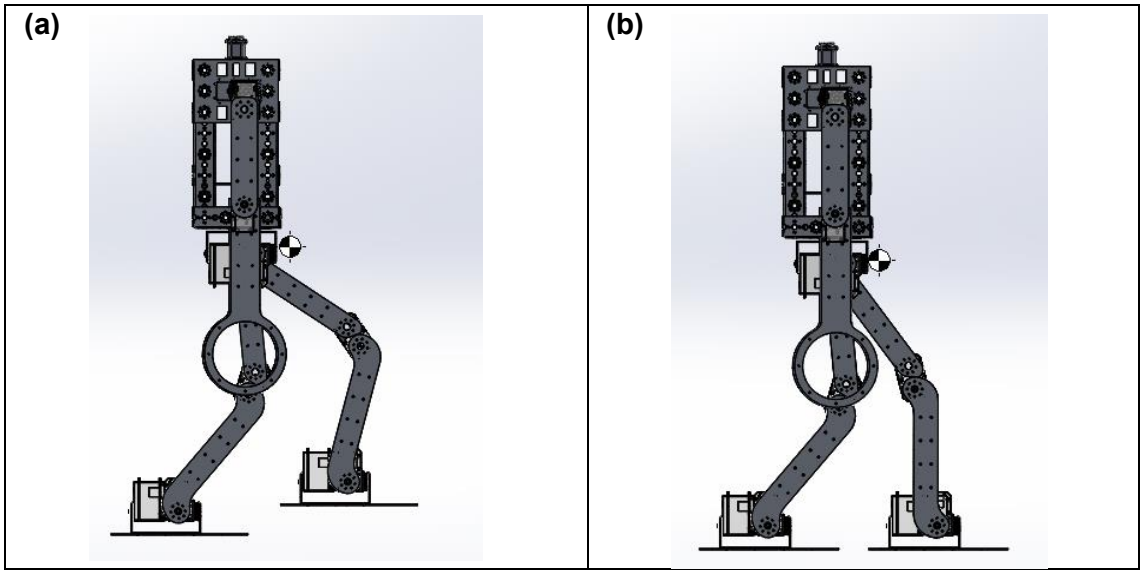


Figura 2.1 – Fases de Apoio: (a) Fase de simples apoio (SSP); (b) Fase de duplo apoio (DSP) (VENÂNCIO, 2016).

2.1.2 Polígono de apoio

Para os seres humanos, equilibrar-se em duas pernas é intuitivamente mais fácil do que em apenas uma, e essa sensação pode ser explicada pelo conceito de Polígono de Apoio. Este termo refere-se ao invólucro convexo que envolve a projeção dos pontos de apoio do bípede na superfície à qual ele se encontra (KAJITA *et al.*, 2014).

Quando os dois pés do bípede estão no chão, ou seja, na fase DSP, o polígono de apoio corresponde ao polígono que compreende ambos os pés, Figura 2.2(a). Para o caso intermediário, Figura 2.2(b), em que um dos pés está deixando o chão, o polígono de apoio compreende toda a região dos pés que está em contato com o chão. Para a fase de simples apoio, o polígono de apoio passa a ser exatamente o formato do pé que está em contato com o chão.

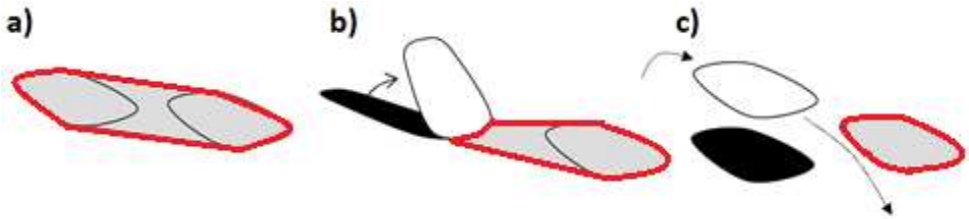


Figura 2.2 – Exemplos de Polígonos de Apoio (Contorno em vermelho): (a) Na fase de duplo apoio; (b) Parcialmente na fase de duplo apoio; (c) Na fase de simples apoio. Adaptado de (EINDHOVEN, 2009).

Pode-se afirmar que quanto maior a região deste polígono de apoio, maior a probabilidade de o bípede manter o equilíbrio, e, portanto, justifica a intuição de que manter-se sobre dois pés é mais confortável em termos de equilíbrio do que em apenas um.

2.1.3 Tipos de Caminhada

Institivamente, ao caminhar sobre uma superfície escorregadia, o ser humano tende a caminhar de forma cautelosa, com movimentos lentos, que tendem a deixá-lo em equilíbrio mesmo se parar de se mexer instantaneamente. Já na caminhada comum, em um ambiente plano como uma calçada, um indivíduo caminha de forma rítmica, com velocidade maior que no caso anterior e sem tanta preocupação com a possibilidade de paradas repentinas. O primeiro caso trata-se da caminhada estática, em que o bípede mantem a projeção do seu centro de massa sobre o seu polígono de apoio durante toda sua locomoção, Figura 2.3(a). Já o segundo caso, caracteriza a caminhada dinâmica, em que a pessoa mantem o seu equilíbrio mesmo em momentos onde a projeção do centro de massa no chão esteja fora do polígono de apoio, Figura 2.3(b).

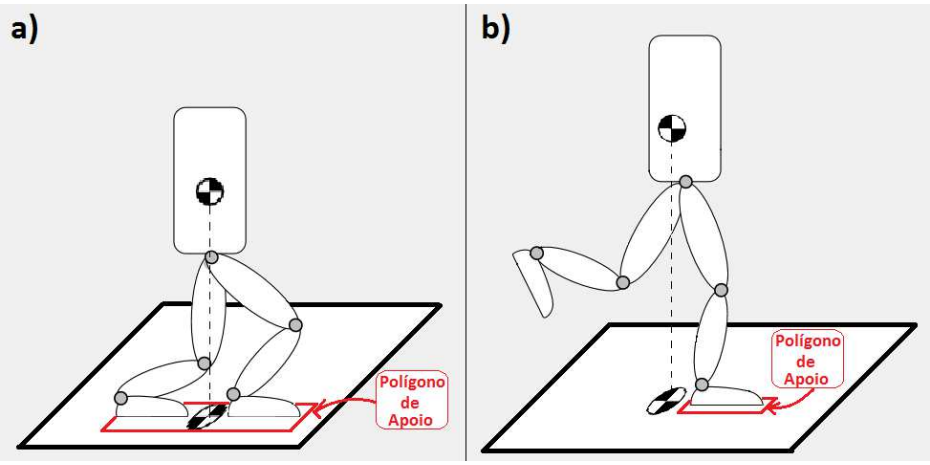


Figura 2.3 - Tipos de caminhada: (a) Estática; (b) Dinâmica (VENÂNCIO, 2016).

Devido à rapidez de locomoção e desafios em manter a caminhada dinâmica, a literatura possui uma gama maior de estudos nessa situação (KAJITA *et al.*, 2014).

2.1.4 Critério de Estabilidade do Ponto de Momento Zero (ZMP)

Na história das pesquisas relacionadas à caminhada bípede, um dos trabalhos publicados de maior impacto até os dias atuais foi o de Vukobratovic e Stepanenko (1972), onde desenvolveu-se um critério para a avaliação da estabilidade dinâmica de um robô

humanoide durante a realização do seu movimento. Este conceito é conhecido como ZMP (do inglês *Zero-Moment Point*, ou Ponto de Momento Zero).

Além dos graus de liberdade ativos que um robô humanoide normalmente possui, existe ainda um grau de liberdade adicional, que tem característica passiva e está localizado entre o chão e o pé do robô. Este grau de liberdade tem grande importância para seu equilíbrio, uma vez que não é desejado que o robô gire em direção do chão, ou seja, que ele caia durante o seu movimento de caminhada. Portanto, a capacidade de manter a estabilidade durante a locomoção está relacionada à capacidade de controlar a ação deste GDL (grau de liberdade) passivo, e isso só pode ser feito indiretamente, através do controle da dinâmica da interferência e/ou controle na dinâmica do robô, através da escolha de trajetórias que atendam ao ZMP (VUKOBRATOVIĆ; BOROVAC, 2004).

Fisicamente, este ponto de momento zero está presente no polígono de apoio do humanoide, e é definido como o ponto onde as componentes horizontais de momento das forças de reação no chão são iguais a zero (KAJITA *et al.*, 2014). As Figura 2.4(a) e (b) ilustram a distribuição de forças no pé de um robô humanoide para a situação 2D. É possível observar que existem tanto as forças de reação verticais, $\rho(\xi)$, quanto horizontais, $\sigma(\xi)$ que variam ao longo do pé de acordo com ξ, que é uma distância intermediária entre os pontos que limitam o pé: x_1 e x_2. Essa distribuição de forças pode ser simplificada nas forças f_x, f_z e um torque τ para um dado ponto arbitrário p_x entre x_1 e x_2, Figura 2.4(c). O ZMP é justamente (na situação 2D) o ponto p_x em que o torque τ é nulo. O polígono de apoio na perspectiva 2D mostrado na Fig. 2.4(c) é justamente a reta entre os pontos x_1 e x_2, ou seja, é mostrado apenas uma secção do polígono. Caso o ponto p_x tenha uma resultante de torque nula ($\tau = 0$) e esteja compreendido entre os pontos x_1 e x_2 durante toda caminhada do humanoide, é garantida a sua estabilidade dinâmica.

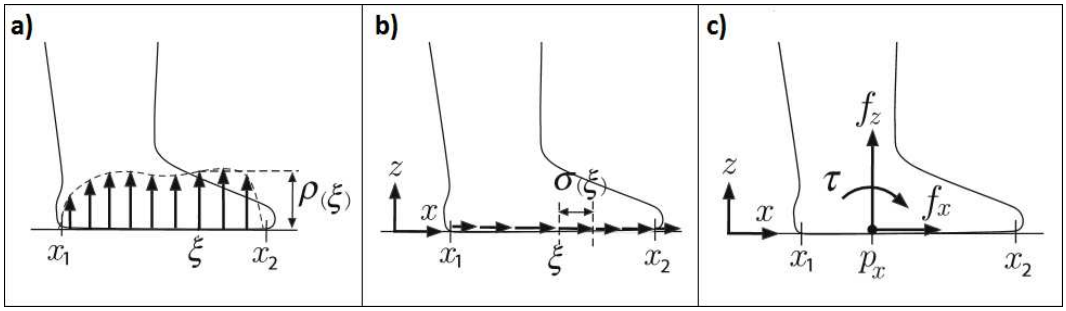


Figura 2.4 - Distribuição das Forças ao longo do pé 2D (a) Forças na vertical $\rho(\xi)$, sendo ξ um valor normalizado entre as extremidades dos pé x_1 e x_2; (b) Forças na horizontal; (c) Resultante das forças de reação f_x e f_z para um determinado ponto p_x do pé (KAJITA *et al.*, 2014).

O conceito de ZMP para duas dimensões pode ser estendido para três dimensões e isso é mostrado em Kajita *et al.* (2014) assim como as demonstrações para o cálculo da posição no espaço cartesiano 3D.

Existem diferentes abordagens para o cálculo da localização do ZMP em relação ao movimento executado. Nesta dissertação, o ZMP será calculado levando em consideração que toda a massa do robô está concentrada no seu centro de massa. Segundo Kajita *et al.* (2014) esse método fornece um valor de ZMP muito próximo ao método que leva em consideração cada segmento do robô separadamente, e ele demonstra que as equações para esta abordagem resultam em:

$$p_x = x - \frac{(z - p_z)\ddot{x}}{\ddot{z} + g} \tag{2.1}$$

$$p_y = y - \frac{(z - p_z)\ddot{y}}{\ddot{z} + g} \tag{2.2}$$

Para um sistema de referência inercial arbitrário em $\mathbb{R}^3$, as variáveis x, y e z correspondem à posição do centro de massa do robô, e consequentemente sua aceleração é dada por $\ddot{x}$, $\ddot{y}$ e $\ddot{z}$. A aceleração da gravidade corresponde à variável g, e a posição do ZMP no plano XY é dado pelas variáveis p_x e p_y. A variável p_z corresponde à altura (ao longo do eixo z) que o polígono de apoio se encontra em relação ao referencial inercial arbitrário adotado.

2.2 Estratégias de Locomoção Bípede

As diferentes estratégias adotadas para a locomoção bípede tratam em sua maioria de uma maneira de gerar trajetórias que satisfaçam a condição de estabilidade do ZMP. Segundo Chiaverini e Meddahi (2015), a cinemática bípede apresenta redundâncias, o que gera mais de uma solução para a correspondência entre a referência do ZMP e os valores que as articulações devem assumir. Dentre as abordagens relacionadas à geração de trajetórias de caminhada humanoide que levam em consideração a dinâmica do sistema, pode-se citar o trabalho de Clever *et al.* (2016) onde o modelo dinâmico do robô humanoide é utilizado para a otimização de trajetórias pré-computadas para a caminhada do robô. Em Missura (2016) um CPG (do inglês *Central Pattern Generator*), foi utilizado como base para o movimento de locomoção para o robô humanoide, e um modelo dinâmico do robô

baseado no pêndulo invertido linear foi utilizado para a implementação de técnicas de controle analíticas, e posteriormente foram otimizadas por técnicas de aprendizado de máquina. Em Liu *et al.*, (2015) é proposto a utilização do modelo Dual-SLIP (do inglês *Dual Spring Loaded Inverted Pendulum*) para a geração de trajetórias estáveis para o robô humanoide, neste modelo, as pernas do robô são vistas como pêndulos invertidos com a adição de uma constante de rigidez, e dessa forma, consegue caracterizar de forma mais completa os efeitos de amortecimento durante a caminhada.

No trabalho de Venâncio (2016) foi mostrado que o método LIPM, proposto por Kajita *et al.* (2014) é capaz de gerar trajetórias estáveis para um ambiente simulado do robô, porém, quando o método é aplicado no robô real, este apresenta instabilidades, que surgem devido às folgas mecânicas das articulações, das imprecisões mecânicas de construção, das limitações de hardware e principalmente da variação das características do terreno. Apesar dessas implicações práticas, o método mostrou-se promissor para a geração online das trajetórias omnidirecionais, além de permitir a modificação online da posição dos pés ao longo da sua caminhada, o que permite uma intervenção de controle.

Assim, para a geração das trajetórias que cada pé do robô deve seguir durante o movimento de caminhada, será aplicada nessa dissertação o método do LIPM. O Capítulo IV traz uma visão geral deste método com algumas modificações realizadas para tornar o método aplicável no robô real.

2.3 Estratégias de estabilização da caminhada

Os métodos citados para a geração da trajetória dos pés não levam em consideração a aplicação de um controle em malha fechada para a estabilização do robô quando sujeito à algum distúrbio externo. Estes métodos geram trajetórias em malha aberta que teoricamente, para um robô livre de distúrbios, promovem um caminhar estável.

Porém, em situações reais, especialmente com a possibilidade de mudança nas características físicas do terreno, é de extrema importância adicionar estratégias que melhorem a qualidade da caminhada humanoide para distúrbios relativamente pequenos quando comparados a aplicações de forças externas. Essas técnicas são denominadas por Kajita *et al.* (2014) de controle estabilizador.

Khadiv *et al.* (2017) realiza a estabilização da caminhada do robô com a modificação da posição dos pés e do tempo de cada passo. Isso é feito com a construção de um gerador de trajetórias que tem restrições lineares, é baseado no LIPM, e tem geração ótima de trajetória a cada ciclo de controle.

Mason *et al.* (2016) baseia o controle do robô na aplicação de um controlador regulador linear quadrático, ou LQR (do inglês *Linear Quadratic Regulator*), que é capaz de determinar um comportamento estabilizador de acordo com a pose do robô e a linearização do seu modelo dinâmico durante a sua movimentação.

Missura e Behnke (2013) reduz a constante de ganho proporcional dos motores da perna do robô, e aplica um controle *feedforward* para compensar efeitos indesejados dessa redução do valor de controle. Essa estratégia é aplicada em um robô humanoide com construção semelhante à utilizada nessa dissertação, e é obtido um comportamento auto estabilizador, em que a baixa rigidez do movimento permite a absorção de impactos durante a troca de pés.

A estratégia de estabilização utilizada nessa dissertação será a de redução do ganho proporcional dos servomotores, pois além de ser relativamente de simples aplicação, não há necessidade que o robô siga perfeitamente as trajetórias propostas, como no caso das técnicas que necessitam de *feedback* de sensores, e torna simples o objetivo de melhoria da caminhada para que este trabalho possa prosseguir com um dos temas centrais que é a identificação de terrenos. O Capítulo V traz detalhes da implementação desta abordagem.

CAPÍTULO III

IMPLEMENTAÇÃO DO *SOFTWARE* E DA SIMULAÇÃO DO ROBÔ HUMANOIDE

Para os testes desenvolvidos neste trabalho foi utilizado tanto um robô real, conforme o apresentado no capítulo introdutório, quanto um robô simulado no *software* Gazebo. Ambos robôs compartilham do mesmo *software*, e a maneira como isso foi elaborado e outros detalhes são abordados neste capítulo.

3.1 Simulação

Um robô simulado foi feito com base no projeto mecânico desenvolvido em *SolidWorks*, Figura 3.1(a). O ambiente de simulação utilizado foi o Gazebo, que tem sido utilizado com frequência no desenvolvimento de robôs em diversos trabalhos (TAKAYA *et al.*, 2016).

A utilização do simulador permitiu o teste de diversos parâmetros de caminhada e ideias durante o desenvolvimento desta dissertação. Além disso, os valores relacionados a torques e posições nas articulações e valores do IMU foram utilizados como indicadores da eficiência dos métodos aplicados, ou seja, permitem avaliar se os movimentos de caminhada, controle propostos e métodos de identificação do terreno tem possibilidade de serem executados no robô real.

O robô simulado é mostrado na Figura 3.1(b), e foi obtido através de um *plugin* para SolidWorks (BRAWNER, 2017) que auxilia na conversão do modelo para um arquivo de texto de extensão URDF (do inglês *Unified Robot Description File*) (YOUSUF *et al.*, 2015), que é o modelo de arquivo utilizado pelo Gazebo e contém as informações das seguintes propriedades mecânicas:

- Relação pai/filho entre cada elemento do robô e suas articulações de ligação com as respectivas posições de centro de massa e formato de cada parte.
- Inércia de cada elemento.

- Massa.
- Coeficiente de atrito.
- Rigidez e amortecimento de cada elemento.

O arquivo URDF contendo o detalhamento do *hardware* do robô é apresentado no Apêndice A.

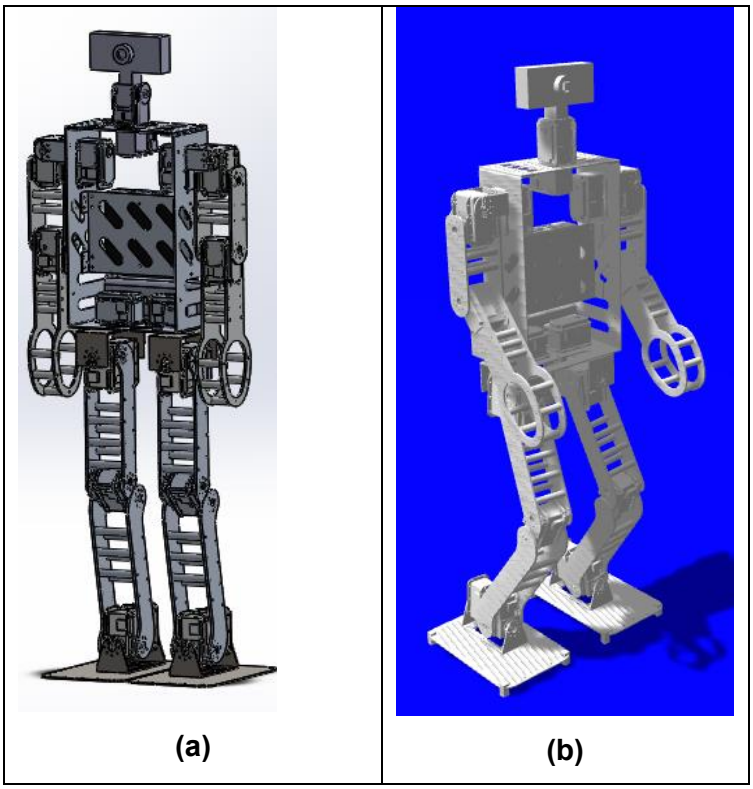


Figura 3.1 – (a) Projeto mecânico da robô Sakura feito no SolidWorks (b) Modelo de simulação do robô no ambiente Gazebo.

O simulador Gazebo permite a configuração de sensores de torque para cada articulação e a inserção de um sensor IMU com características de ruído e precisão configuráveis, para que os valores obtidos sejam semelhantes aos obtidos de sensores reais. Todas essas informações podem ser obtidas e visualizadas através do ambiente ROS instalado no sistema operacional Linux Ubuntu 16.04 LTS. Estas funcionalidades serão detalhadas no subtópico 3.2.

Durante as simulações e os testes no robô real, foi possível apontar de maneira qualitativa alguns comportamentos semelhantes em relação à execução da caminhada proposta. Porém, o robô simulado não apresenta folgas mecânicas e foi justamente esta característica que evidenciou as diferenças entre ambos ambientes.

3.2 *Software* desenvolvido

Todos os algoritmos e métodos mostrados neste trabalho foram aplicados no robô utilizando as linguagens de programação C/C++ e Python. Para o cálculo da posição do centro de massa, da dinâmica inversa e a cinemática direta, utilizou-se a biblioteca RBDL (do inglês *Rigid Body Dynamics Library*), que tem código aberto e desempenho comparável à bibliotecas em estado de arte na área de cálculos cinemáticos e dinâmicos de sistemas multicorpos (FELIS, 2017). Esta biblioteca faz uso do arquivo de descrição do robô (URDF) mencionado anteriormente. Para algumas operações algébricas e manipulação de matrizes e vetores, utilizou-se a biblioteca Eigen, que também tem sólida documentação e é aplicada em diversos projetos (EIGEN, [S.d.]).

Um dos pontos importantes do presente trabalho foi a utilização do ROS (do inglês *Robot Operating System*), que apesar do nome, não se trata de um sistema operacional, mas sim de um conjunto de *frameworks* de *software* para o desenvolvimento em robótica. Esta ferramenta possibilita criar uma infraestrutura de comunicação entre processos que podem comunicar entre si por meio de mensagens, de forma síncrona ou assíncrona (ARAÚJO, 2017).

Nesta dissertação, o *software* final elaborado contempla diversos módulos que desempenham funções específicas, como por exemplo: cálculo da trajetória da caminhada, envio das informações para os servomotores e cálculos da cinemática inversa. Estes módulos foram transformados em *Nodes*, que são os nomes dados aos processos criados no ambiente ROS e que podem comunicar com outros *Nodes* por meio de *Services*, ou *Topics*. Estes dois conceitos são primordiais na comunicação de processos no ambiente ROS.

A modularização em *Nodes* tem a vantagem de permitir que os processos executem de forma paralela e, em um processador com múltiplos núcleos, pode otimizar a utilização dos recursos computacionais. Além disso, o ROS permite que cada processo execute em uma determinada frequência de atualização, e essa frequência pode ser configurada via *software* e é limitada pela velocidade de processamento e aquisição dos dados de um determinado processo. A Figura 3.2 exemplifica uma parte de como é a estrutura do *software* ao utilizar o ROS. Basicamente, o *Topic* é identificado por um nome em que *Node1* publica informações, e tanto o *Node2* quanto o *Node3* leem essas informações, pois estão inscritas neste *Topic*, esta troca de dados é assíncrona. Já o *Service* mostrado na Figura 3.2, também é identificado por um nome, funciona de forma síncrona e tem seu funcionamento baseado em uma solicitação, no caso feita pelo *Node1* que aguarda a resposta do *Node2*.

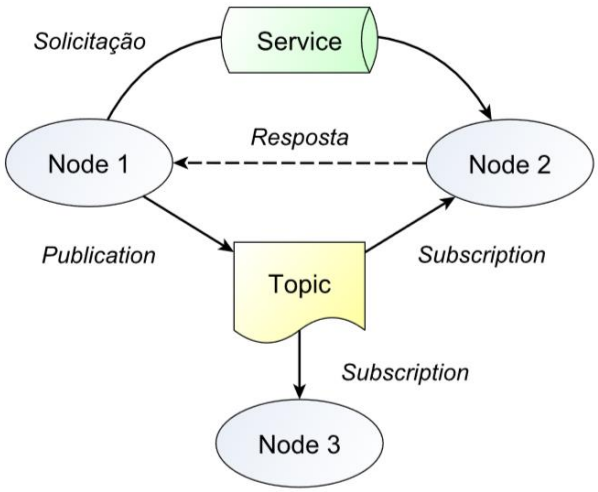


Figura 3.2 – Exemplo de estrutura em *Nodes*, *Topics* e *Services* (ARAÚJO, 2017).

Além das funcionalidades de comunicação entre processos, o ROS possui ferramentas para a visualização de dados em tempo de execução, o *rqt_plot* (WILSON, 2018), que permite uma análise rápida de variáveis importantes do sistema enquanto ele está funcionando.

Outra importante funcionalidade do ROS que foi explorada no *software* desenvolvido é o *dynamic_reconfigure* (SAITO, 2018), que permite a mudança de determinadas variáveis de algum node por meio de uma interface configurável Figura 3.3. Portanto, isso simplifica a tarefa de testar o efeito de alguns parâmetros do sistema durante a execução do mesmo.

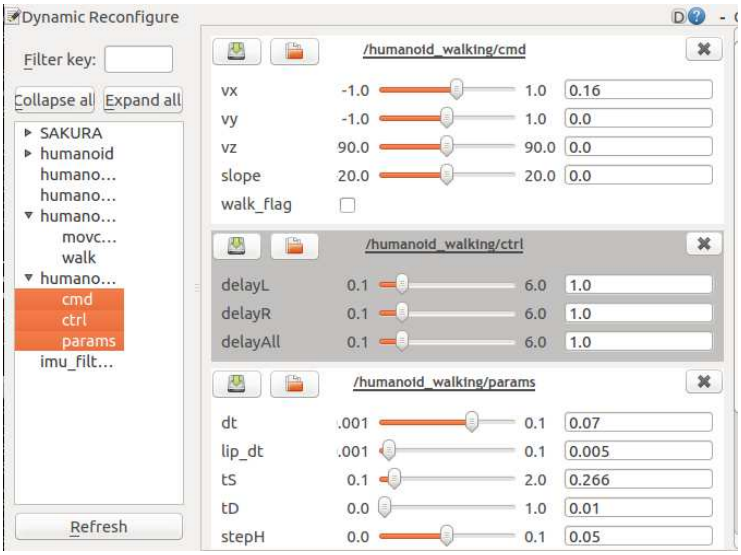


Figura 3.3 – Interface criada para mudar variáveis durante testes tanto com o robô real quanto com o simulado.

O encapsulamento dos processos em *nodes* permitem que estes possam ser utilizados tanto para o robô real quanto para o robô virtual no Gazebo, o que muda entre

cada um é apenas a forma como o processo que foi denominado de *humanoid_interface* se comunica com o hardware.

Para a comunicação com o hardware simulado, utilizou-se o controlador joint_*trajectory_controller* (SABUR, 2017), que basicamente, precisa das informações da posição desejada e do tempo necessário para atingir tal objetivo. O controlador utilizado é o PID e pode ser configurável via software. A comunicação com o robô real foi feita através da biblioteca da fabricante dos servomotores Dynamixel utilizados, neste caso, cada motor necessita que sejam enviadas as informações de posição e velocidade, sendo que esta última é calculada com base no tempo necessário para atingir a posição desejada levando em consideração a última posição enviada. O controle interno também é o PID e pode ser devidamente configurado via software.

Para o tratamento do sinal do sensor IMU utilizou-se o filtro *Madgwick* (MADGWICK, 2010), que trata os sinais ruidosos dos três eixos do acelerômetro e giroscópio e obtém a orientação tridimensional do sensor. Esta informação é dada em *quaternion* e pode ser convertida para ângulos de Euler para facilitar a análise da inclinação do robô enquanto executa os movimentos propostos. A implementação do filtro foi feita através do pacote *imu_tools* (GUENTHER, 2015), que já possui a implementação do algoritmo do filtro em forma de *Node*, o que facilita a integração com o restante do código.

A estrutura geral do programa desenvolvido nesta dissertação é apresentada na Figura 3.4. As entradas são definidas pelo usuário do programa, e são configuradas através da janela de configuração apresentada, Figura 3.3. Estas entradas estão relacionadas com a velocidade desejada para o robô andar para frente, v_x, em metros por segundo, os valores de parâmetros desejados para o controle PID que são adimensionais e para a configuração da caminhada, como o tempo de simples apoio T_s em segundos, tempo de duplo apoio T_d, também em segundos, e a altura do centro de massa z_c do robô em metros. Estas informações são encaminhadas para os *Nodes* que as utilizam para executar suas respectivas funções.

O bloco de geração de trajetórias, utiliza as equações relacionadas à dinâmica do pêndulo invertido para a geração de trajetórias cartesianas que cada pé deve alcançar a fim de atingir o objetivo de caminhar. O Capítulo IV será responsável por explicar em detalhes o bloco de geração de trajetórias. Em seguida, essas coordenadas são convertidas para valores de ângulos que correspondem às posições que as articulações devem atingir para realizar o movimento proposto pelo módulo geração de trajetórias. A forma de obter estes ângulos com base na cinemática inversa é mostrada tanto no Capítulo IV, quanto no Apêndice B. É neste módulo que é obtido a posição do centro de massa do robô com base no modelo URDF, e essa coordenada é repassada para o módulo anterior de geração de

trajetórias, e isso é feito já na inicialização do programa.

As informações angulares geradas pelo módulo de cinemática inversa são convertidas em mensagens adequadas para o robô que estiver conectado, ou seja, é nessa parte que se estabelece a comunicação com o meio desejado. Neste mesmo módulo é realizado a leitura dos sensores IMU, torque, posição atual dos servomotores e é feito o tratamento do sinal, como no caso do sensor inercial. Além disso, este módulo é responsável por fazer a leitura dos valores de parâmetros PID desejados e enviar para os motores. O Capítulo V trará em detalhes como que essa escolha dos parâmetros PID foi feita para a estabilização da caminhada do robô.

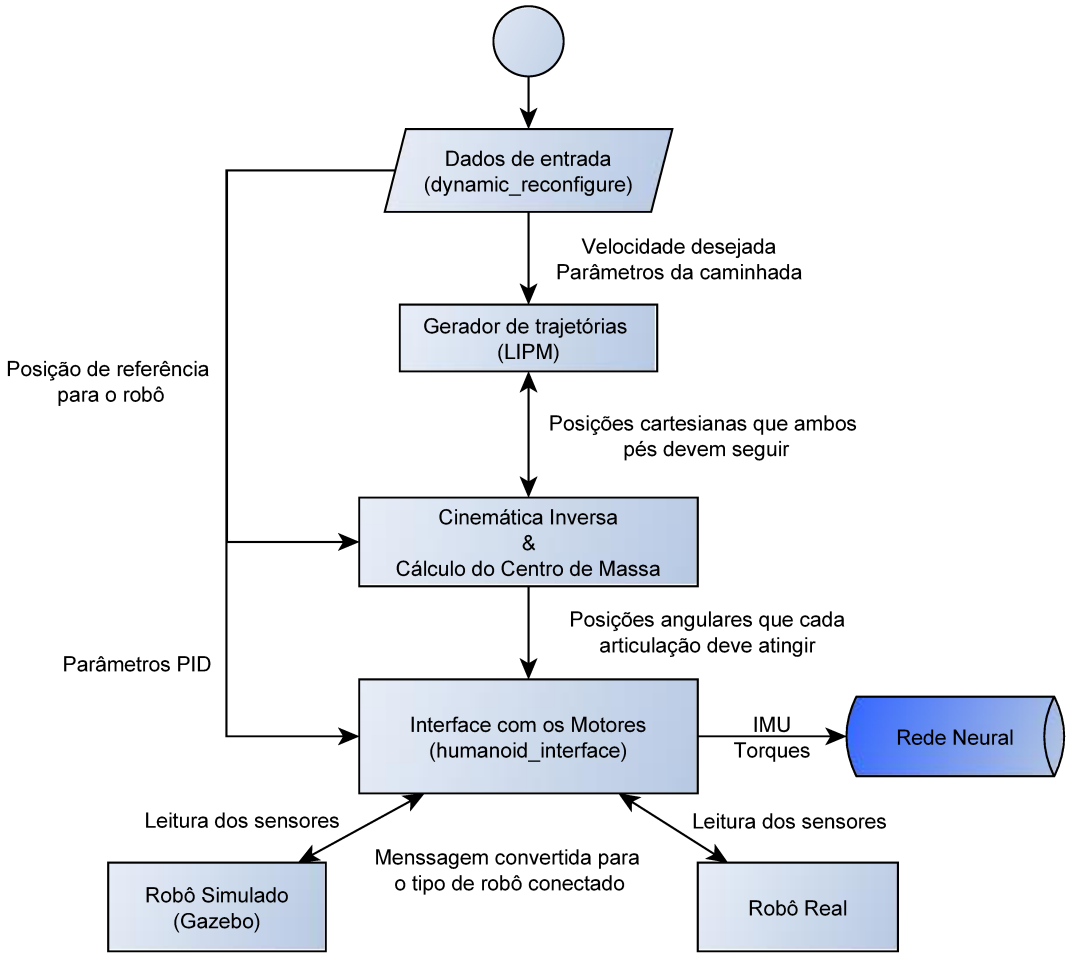


Figura 3.4 – Visão geral do software desenvolvido nesta dissertação.

Por fim, os dados de saída dos sensores de IMU e de torque obtidos durante a caminhada pelo Node de interface com os servomotores, são encaminhados para o *Node* Rede Neural que é programado em Python, e é responsável pelo treinamento da rede neural e pela execução dessa rede após o treino. A saída desse *Node* é uma informação estatística

das chances de que cada terreno tem de ser o atual ambiente em que o robô está se locomovendo.

Para a programação da rede neural foi utilizado a biblioteca TensorFlow (ABADI *et al.*, 2016) em conjunto com a TFLearn (TANG, 2016). O funcionamento desse *Node*, será detalhado no Capítulo VI.

Todos os códigos utilizados para a programação destes *Nodes*, assim como as informações relacionadas à forma de instalação em ambiente Linux, podem ser encontrados no link: https://github.com/murilomend/humanoid_movement.

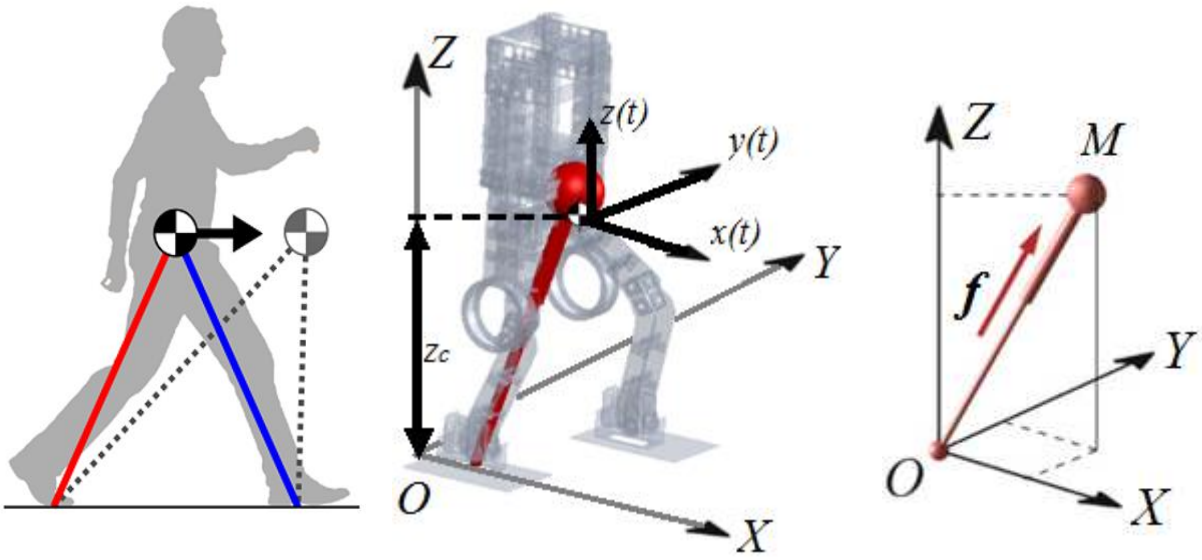
Com o software esquematizado pretende-se obter uma caminhada para o robô humanoide que seja estável de acordo com os parâmetros de controle e de caminhada selecionados, mesmo com mudanças de rigidez de terrenos, Capítulo V.

Além disso, através dos dados obtidos para análise, será efetuado o processamento das informações dos sensores para caracterização do tipo de terreno em que o robô se encontra, Capítulo VI.

CAPÍTULO IV

MÉTODO DO PÊNDULO INVERTIDO LINEAR

O método LIPM (*Linear Inverted Pendulum Method*), utiliza basicamente a modelagem dinâmica de um pêndulo invertido para representar a dinâmica do robô. Toda a massa do robô é concentrada em seu CM (Centro de Massa), e a perna de apoio funciona como o pêndulo invertido que deve fazer o trabalho de manter a altura do CM constante, Figura 4.1.



(a) **(b)** **(c)**

Figura 4.1 – Ilustrações do princípio do método do pêndulo invertido linear: (a) Ilustração para o corpo humano em 2D; (b) Situação 3D em um robô humanoide; (c) Modelo simplificado da dinâmica do pêndulo em 3D, sendo M a massa total do humanoide concentrada na extremidade do pêndulo, e $\boldsymbol{f}$ é uma força aplicada nessa massa (VENÂNCIO, 2016).

Através de uma modelagem matemática simplificada do robô é possível obter as equações do movimento para o centro de massa de acordo com as coordenadas de onde o

robô está designado a pisar. Sendo $x(t)$ a posição do centro de massa na direção que aponta para frente do robô, e $y(t)$ a coordenada lateral do CM, para uma altura constante do centro de massa z_c, Fig. 4.1(b). As seguintes equações descrevem o movimento do centro de massa de acordo com as condições iniciais:

$$x(t) = x(0)\cosh\left(\frac{t}{T_c}\right) + T_c\dot{x}(0)\sinh\left(\frac{t}{T_c}\right) \tag{4.1}$$

$$\dot{x}(t) = \frac{x(0)\sinh(t/T_c)}{T_c} + \dot{x}(0)\cosh\left(\frac{t}{T_c}\right) \tag{4.2}$$

$$y(t) = y(0)\cosh\left(\frac{t}{T_c}\right) + T_c\dot{y}(0)\sinh\left(\frac{t}{T_c}\right) \tag{4.3}$$

$$\dot{y}(t) = \frac{y(0)\sinh(t/T_c)}{T_c} + \dot{y}(0)\cosh\left(\frac{t}{T_c}\right) \tag{4.4}$$

$$z(t) = z_c \tag{4.5}$$

$$\dot{z}(t) = 0 \tag{4.6}$$

$$T_c = \sqrt{\frac{z_c}{g}} \tag{4.7}$$

Estas equações são deduzidas em Kajita *et al.* (2001), e é considerado que altura z_c do centro de massa é constante durante toda caminhada, e por isso sua velocidade deve ser nula ($\dot{z}(t) = 0$). A variável T_C é uma constante de tempo que determina a duração de cada passo executado pelo robô ao longo do tempo t. Nas Eqs. (4.1) a (4.7) *cosh* representa o cosseno hiperbólico e *sinh* o seno hiperbólico.

Este método é totalmente detalhado em Kajita *et al.* (2001), e pode-se dizer que a ideia é fornecer as posições que os pés devem atingir, e através das Eqs. (4.1)-(4.7), obter as coordenadas que o centro de massa do robô deve passar para cada instante de tempo. A Figura 4.2 ilustra essa abordagem, em que as variáveis S_y e $S_x(n)$, determinam, respectivamente a distância que os pés devem estar durante a caminhada, e o comprimento do n-ésimo passo a ser efetuado. Dessa forma, com os valores de S_y e $S_x(n)$ é possível obter as condições iniciais de posição ($x(0)$ e $y(0)$) para as Eqs. (4.1) e (4.3) por onde o centro de massa do robô deve passar. Ao determinar que o robô parte do repouso com velocidades iniciais nulas, ou seja, que para o passo $n = 0$, $\dot{x}(t) = 0$ e $\dot{y}(t) = 0$, é possível obter as velocidades iniciais do centro de massa para o passo $n + 1$ com as velocidades finais atingidas pelas Eqs. (4.2) e (4.4), e isso pode ser feito sucessivamente durante a caminhada, fazendo que o movimento de locomoção seja gerado continuamente com as escolhas das posições dos pés através de $S_x(n)$ e S_y.

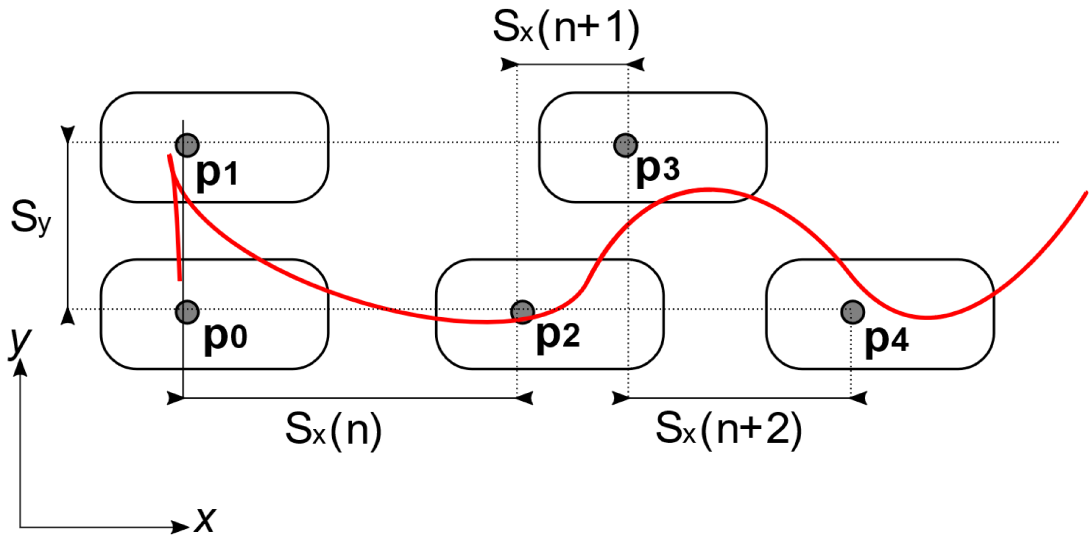


Figura 4.2 – Ilustração do funcionamento do método de geração de trajetórias do pêndulo invertido linear: as posições dos pés são determinadas pelos valores da distância entre os passos ($S_x(n)$ e S_y), e então a posição do centro de massa, representado pela curva vermelha que passa entre os pés, é determinado pelas Eqs. (4.1)-(4.7). O valor de *n* corresponde ao n-ésimo passo p_n (VENÂNCIO, 2016).

A variável T_s é introduzida como o tempo de simples apoio (relacionado ao SSP), em que determina o tempo necessário para que o pé saia do chão e retorne. Com base nessa variável, e nas distâncias desejadas para cada passo (com S_x e S_y), as condições iniciais para as Eqs. (4.1)-(4.4) podem ser determinadas para cada passo dado.

Além disso, o conceito de tempo de duplo apoio (DSP) é introduzido por Kajita *et al.*, (2014), e é adicionado ao LIPM entre duas fases de simples apoio Figura 4.3, onde a posição do centro de massa nesse caso é determinado por um polinômio interpolador que tem como condições iniciais finais determinadas por $SSP_n(T_s + T_d)$ e $SSP_{n+1}(T_d)$, em que T_d é o tempo de duplo apoio inserido. Ao comparar o efeito da inserção dessa fase com o caso em que não houve inserção, Figura 4.4, é possível perceber que o período de duplo apoio tem o potencial de reduzir a descontinuidade na velocidade em comparação com uma caminhada puramente de simples apoio, e consequentemente, essa estratégia reduz possíveis instabilidades causadas por picos de aceleração.

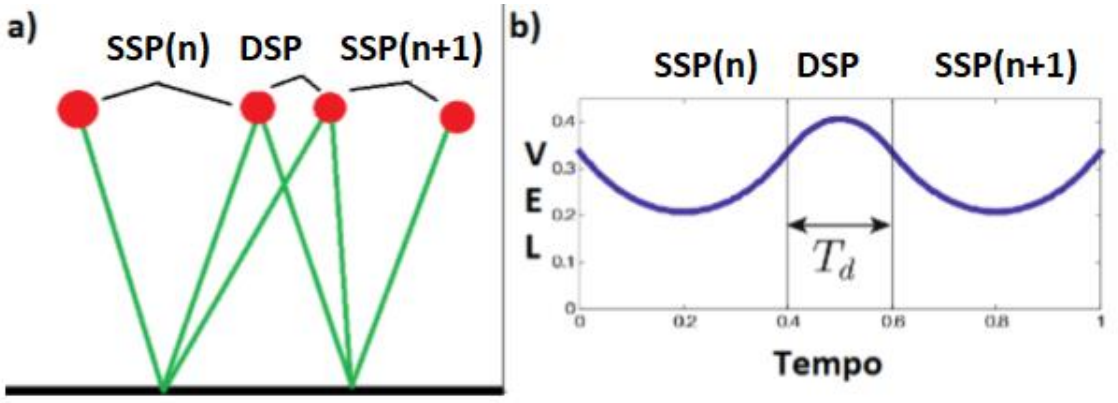


Figura 4.3 – (a) Ilustração da transição do pêndulo, entre duas fases de simples apoios (SSP) com uma fase intermediária de duplo apoio (DSP). (b) Efeito na velocidade do centro de massa do robô para caminhada com a fase de duplo apoio (DSP), observa-se que o efeito da descontinuidade foi minimizado com uma curva intermediária de velocidade (VENÂNCIO, 2016).

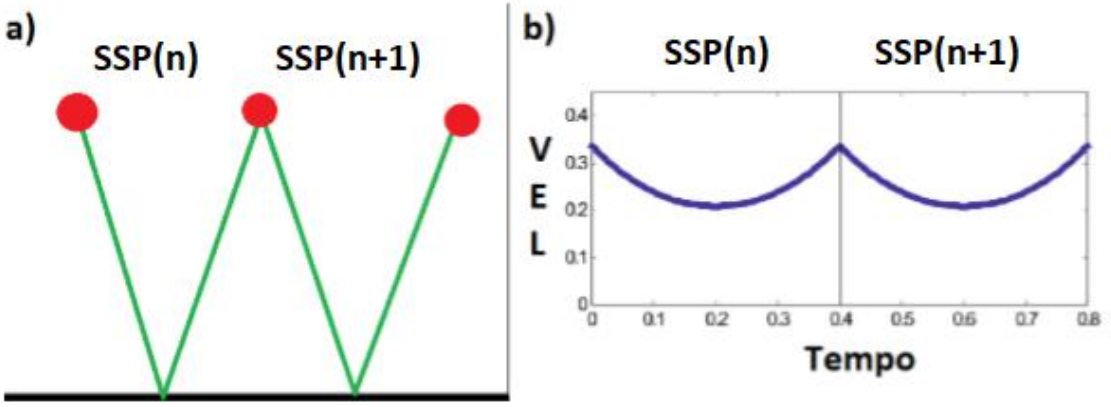


Figura 4.4 – (a) Ilustração da transição do pêndulo, que no caso do robô seria uma mudança de pernas entre duas fases de simples apoios (SSP). (b) Efeito na velocidade do centro de massa do robô para caminhada sem a fase de duplo apoio (DSP), observa-se que há uma descontinuidade no momento de transição de perna de apoio (VENÂNCIO, 2016).

Venâncio (2016) utiliza um polinômio de segundo grau para realizar a interpolação entre os dois momentos SSP com a justificativa de manter a aceleração constante e não ter a possibilidade de gerar o ponto ZMP fora do polígono de apoio do robô. Essa mesma estratégia foi mantida nesta dissertação.

É importante notar que o LIPM não determina o movimento vertical que o pé deve executar entre um passo e outro, e para isso, foi parametrizada uma curva senoidal elevada ao quadrado. Esse perfil foi obtido empiricamente depois de testes tanto na simulação, quanto no robô real, e demonstrou uma suavização do movimento de levantar o pé, e

desestabilização menor que uma curva puramente senoidal. A Figura 4.5 ilustra como uma curva escolhida é aplicada durante o movimento no ar do pé do robô.

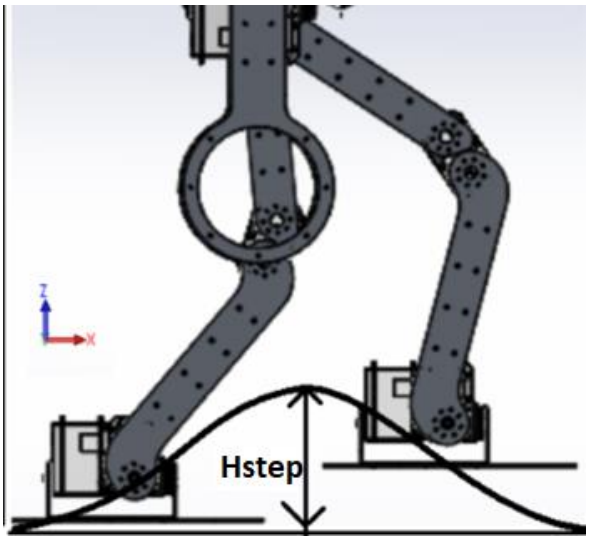


Figura 4.5 – Ilustração da trajetória do pé durante o movimento de levantar e descer durante um passo. A curva escolhida é uma senoidal ao quadrado e tem amplitude definida por H_{step}. Reproduzido de (VENÂNCIO, 2016).

A altura máxima que o pé atinge durante o movimento vertical é determinada pela variável H_{step} e pode ser configurada conforme a necessidade. Durante os testes tanto no robô real quanto no robô simulado, verificou-se que o aumento do valor dessa variável introduz instabilidades ao movimento, e no caso do robô real, a redução desse valor pode causar constantes colisões do pé do robô com a grama sintética, ou mesmo com o chão por conta das folgas mecânicas.

As variáveis utilizadas para a configuração do LIPM nesta dissertação são: o tempo de simples apoio T_s, o tempo de duplo apoio T_d, a altura do centro de massa do robô z_c e a altura do passo H_{step}. Matematicamente, pode-se formular a seguinte função para o LIPM:

$$\begin{bmatrix} \boldsymbol{P_{dir}} \\ \boldsymbol{P_{esq}} \\ \boldsymbol{CM_{pos}} \end{bmatrix} = \boldsymbol{LIPM}(T_s, T_d, z_c, H_{step}) \tag{4.8}$$

Onde $\boldsymbol{P_{dir}}$ e $\boldsymbol{P_{esq}}$, denotam respectivamente a posição cartesiana no espaço 3D dos pés direito e esquerdo, e da mesma forma $\boldsymbol{CMpos}$ para o centro de massa.

Com essa função, é possível gerar trajetórias para o centro de massa e para os pés que são teoricamente estáveis para qualquer configuração de caminhada. A Figura 4.6

mostra a caminhada no plano XY com a posição do centro de massa para os parâmetros listados na Tabela 4.1. Pode-se observar que o aumento do valor de simples apoio T_s gera o aumento da amplitude do movimento, Fig. 4.6(b), o que é justificado pela necessidade de o robô deslocar o seu centro de massa para o ponto da articulação do pé, de forma a satisfazer a condição de estabilidade ZMP, conforme pode ser verificado na Fig. 4.7. A redução do valor de T_s para uma mesma velocidade v_x tende a acelerar o centro de massa, e consequentemente, a amplitude do movimento ao longo do eixo Y é menor, e como no caso anterior obedece ao critério de ZMP.

Tabela 4-1 – Valores utilizados para o teste do modelo LIPM

	T_s [s]	T_d [s]	v_x [m/s]	z_c [m]
Teste-1	0.4	0.03	0.12	0.40
Teste-2	0.8	0.03	0.12	0.40

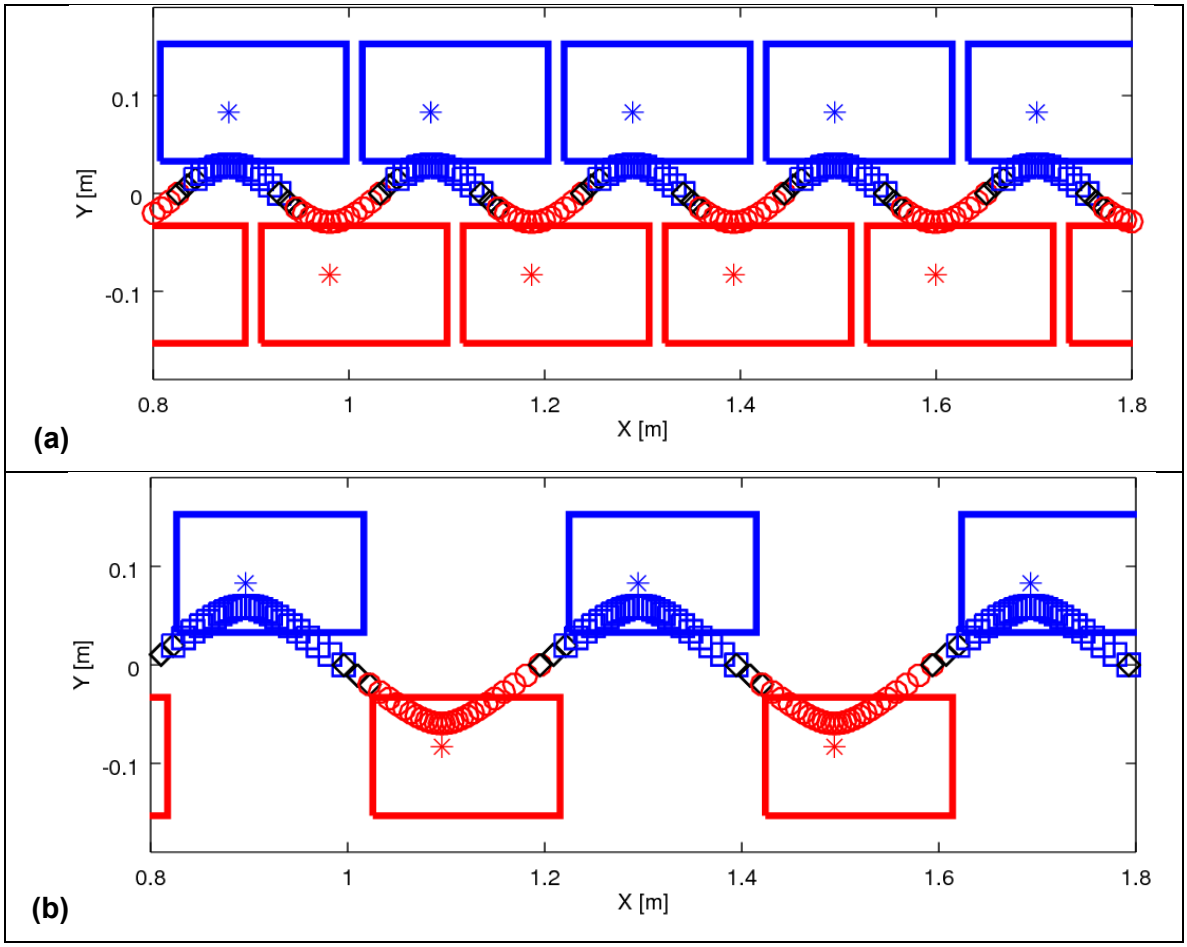


Figura 4.6 – Dois testes realizados para configurações diferentes do LIPM, Tabela 4.1. Cada retângulo simboliza um pé, sendo o pé direito em vermelho, e o pé esquerdo em azul. Os

marcadores quadrados azuis indicam a posição do centro de massa durante o momento da fase de simples apoio com o pé esquerdo, da mesma forma os de formato circular vermelhos são para o pé direito, e a posição do CM para a fase de duplo apoio é simbolizada pelos losangos pretos. (a) Posição do centro de massa para o Teste-1. (b) Posição do centro de massa para o Teste-2.

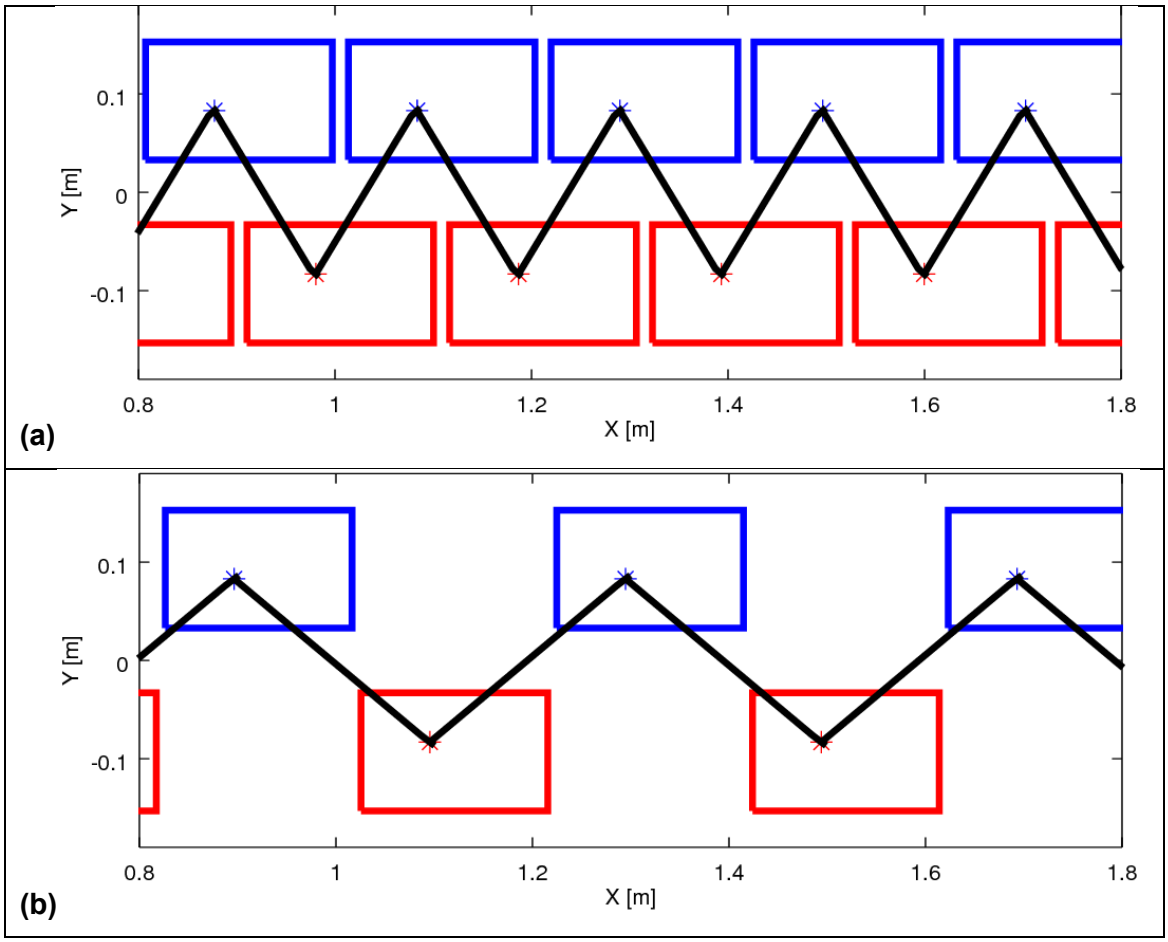


Figura 4.7 – Resultados referentes à posição do ZMP, representado pela linha preta contínua, ao longo da caminhada para os dois testes com as configurações mostradas na Tab. 4.1. (a) Posição do ZMP para o Teste-1. (b) Posição do ZMP para o Teste-2.

Esses resultados são satisfatórios para o caso de um robô ideal em que tem sua dinâmica muito próxima à de um pêndulo invertido de fato. Porém, para fazer com que essas trajetórias sejam utilizáveis no robô tanto simulado, quanto o real, algumas variáveis e técnicas devem ser implementadas nesses casos.

4.1 Mudanças no LIPM

O método LIPM já teve o seu funcionamento comprovado em diversos trabalhos de robótica humanoide (KAJITA *et al.*, 2003), (DALEN, 2012). É importante notar que uma das características comum entre os trabalhos citados é a robustez dos robôs humanoides, que contam com servomotores de alta resolução, com folga mecânica reduzida e ainda apresentam torque suficiente para os movimentos de locomoção. Estes atributos facilitam a correta execução das trajetórias calculadas pelo método do pêndulo invertido.

O robô utilizado neste trabalho apresenta características que dificultam a aplicação de técnicas de locomoção que dependem de uma execução precisa das trajetórias calculadas. Dentre os obstáculos, estão às folgas que podem chegar a 15 graus em relação à posição desejada para a articulação, dependendo da configuração que o robô se encontra. Além disso, o próprio controle PID utilizado no nível de articulações é responsável por inserir erros no sistema, uma vez que ele não é capaz de rejeitar todos os efeitos não lineares devidos aos acoplamentos entre articulações, e esse erro pode variar conforme as constantes de controle escolhidas, como será melhor explicado no Capítulo V. Foi observado que o conjunto dessas perturbações listadas geram basicamente dois efeitos indesejados para o controle de posição do sistema:

- Efeito passa-baixa: a redução dos valores de ganho proporcional pode gerar tanto um atraso na fase quanto uma redução da amplitude desejada para o movimento de cada articulação. Esse mesmo efeito é observado por Schwarz e Behnke (2013), onde é proposto um controle *feedforward* para a compensação desses efeitos através de uma planta do motor que considera efeitos de atrito nas engrenagens do mecanismo.
- Movimentos com amplitude maior que o desejado: a folga de cada articulação permite que a rotação entre os membros siga a inércia na região que o controle não consegue atuar.

Observa-se que ambos efeitos são de natureza não-linear e de difícil modelagem para compensá-los via alguma técnica de controle. Para tentar reduzir tais distúrbios, introduziu-se novas variáveis para o modelo LIPM que atuam na velocidade de execução do movimento, o que possibilita corrigir alguns problemas de amplitude de movimento.

Segundo o modelo LIPM para uma dada altura de centro de massa z_c, tempo de simples apoio T_s e de duplo apoio T_d é calculado a trajetória que o centro de massa deve executar no plano paralelo ao chão. A amplitude do movimento lateral é definida basicamente por essas variáveis, porém, no robô real foi observado que o efeito indesejado da folga faz com que esta amplitude seja maior que a referência calculada. Em outras

palavras, é como se o robô tivesse executando uma trajetória LIPM diferente daquela desejada.

Com o objetivo de poder encontrar um equilíbrio entre o desejado e o executável pelo robô, foi adicionado uma variável γ capaz de acelerar, ou desacelerar o movimento independente dos parâmetros de entrada do sistema, ou seja, é possível configurar uma caminhada com um determinado tempo T_s e fazer com que o robô execute o movimento com um tempo $T_s \times \gamma$, porém com a mesma amplitude do movimento T_s original. Dessa forma, é possível fazer um ajuste com o objetivo de compensar a folga lateral, que se trata do movimento angular indesejado que é somado à posição angular solicitada para os motores que giram em torno do eixo X. Para valores de γ maiores que um, o movimento se torna maios lento, valores entre zero e um, tornam o movimento mais rápido, e o valor unitário não causa nenhuma modificação no movimento original. Esse método mostrou-se efetivo para que o movimento fosse realizado no robô real.

4.2 Mudanças na postura do robô

O método do pêndulo invertido linear fornece as posições no tempo que o pé deve estar em relação ao centro de massa para que a caminhada do robô obedeça ao critério de estabilidade ZMP. Esta abordagem assume que os pés do robô estejam sempre paralelos ao chão, e a posição dos braços não são diretamente determinadas. Além disso, é considerado que o robô apresente sempre distribuição de massa e capacidades mecânicas simétricas, ou seja, não são consideradas possíveis desbalanços de massa, ou mesmo que haja folga mecânica de um lado mais que o outro. A posição dos braços não é diretamente obtida pelo modelo, e, portanto, deve ser estabelecida conforme necessário. Portanto, para contornar essas limitações do LIPM, foi considerado um vetor complementar, Eq. 4.9, para a configuração da postura de referência do robô, que fornece um grau de liberdade maior para ajustar o movimento ao robô conforme necessidade. Essas mudanças foram necessárias para a adequação do método para o robô real, em que foi encontrada uma postura, onde os efeitos das folgas mecânicas e assimetrias fossem minimizados.

$$\boldsymbol{B} = \begin{bmatrix} Incl \\ SideIncl \\ Squat \\ LegOpen \\ FootIncl \\ ComX \\ ComY \\ Arm0 \\ Arm1 \\ Arm2 \end{bmatrix} \tag{4.9}$$

A Figura 4.7 mostra o efeito dessas variáveis na postura do robô. A variável $Incl$ tem o efeito de inclinar o robô para frente ou para trás, e isso possibilita deslocar o centro de massa do robô conforme a necessidade de manter o equilíbrio.

A variável $SideIncl$, tem a propriedade de inclinar o robô lateralmente, e isso pode ser útil para compensar assimetrias da montagem mecânica do robô, principalmente para o robô real.

A variável $Squat$, está relacionada ao agachamento do robô, ela determina a distância que o pé está do quadril ao longo do eixo Z, e, portanto, pode deslocar o centro de massa verticalmente conforme for necessário.

A distância entre as pernas é determinada pela variável $LegOpen$, e tem influência na estabilidade lateral do movimento.

A variável $FootIncl$ determina a inclinação do pé do robô de forma a deixar o robô mais inclinado para frente ou para trás, e tem efeito comparável com o $Incl$.

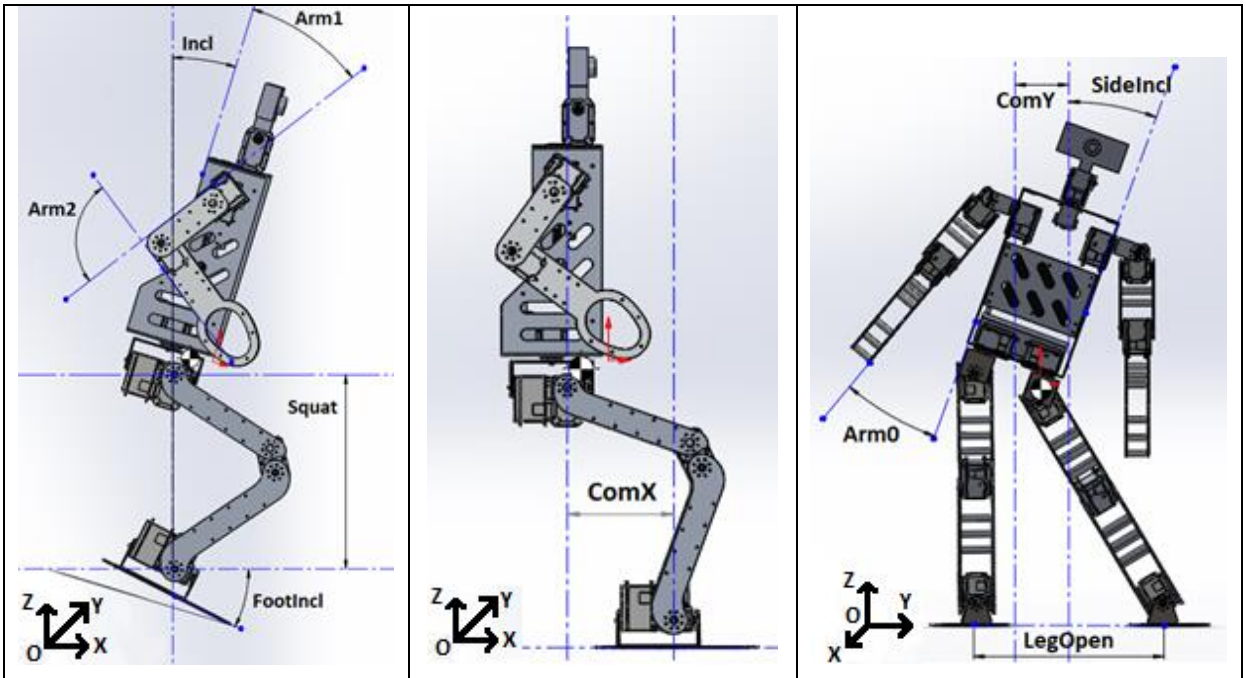
As variáveis $ComX$ e $ComY$, deslocam o quadril do robô no eixo X e Y, respectivamente, e isso é feito sem interferir na inclinação da parte superior do corpo.

As variáveis $Arm0$, $Arm1$ e $Arm2$ correspondem aos valores de ângulos desejados para os três graus de liberdade de cada braço, e conforme a configuração desses valores, os braços podem assumir posições que interfiram no posicionamento do centro de massa do robô e, consequentemente, no seu equilíbrio. Essas variáveis são processadas no cálculo da cinemática inversa do robô. O modelo geométrico inverso utilizado é o mesmo apresentado por Kajita *et al.* (2014), onde estão todas as deduções e equações. Basicamente, a função de cinemática inversa sem a consideração do vetor $\boldsymbol{B}$, tem como entrada as posições dos pés em relação ao ponto central do quadril e os valores de comprimento das pernas. Matematicamente, a cinemática inversa obtém a seguinte relação:

$$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\end{bmatrix} = CinemáticaInversa(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{W}) \tag{4.10}$$

Onde os valores do vetor composto pelos seis valores angulares θ_i são as rotações necessárias para a configuração de uma das pernas do robô, o parâmetro $\boldsymbol{Q}$ denota a posição (q_x, q_y, q_z) e a orientação desejada para o quadril $(q_\theta, q_\phi, q_\gamma)$, da mesma forma o parâmetro $\boldsymbol{P}$ define posição e orientação para um dos pés e o parâmetro $\boldsymbol{W}$ representa os parâmetros relativos à perna, como o tamanho do "fêmur", da "tíbia", e à qual perna deve ser calculada a cinemática inversa. As rotações (θ, ϕ, γ) estão em ângulos de Euler e representam as rotações em torno dos eixos x, y e z respectivamente.

$$\boldsymbol{P} = \begin{bmatrix}p_x\\ p_y\\ p_z\\ p_\theta\\ p_\phi\\ p_\gamma\end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix}q_x\\ q_y\\ q_z\\ q_\theta\\ q_\phi\\ q_\gamma\end{bmatrix} \text{ e } \boldsymbol{W} = \begin{bmatrix}-1\ para\ perna\ esquerda\ e\ +1\ para\ perna\ direita\\ Comprimento\ do\ Fêmur\\ Comprimento\ da\ Tíbia\\ Distância\ do\ início\ da\ perna\ com\ o\ meio\ do\ corpo\end{bmatrix} \tag{4.11}$$



(a)	(b)	(c)

Figura 4.8 – Influência dos parâmetros introduzidos pelo vetor $\boldsymbol{B}$ na postura do robô. (a) Variáveis $Incl$, $Squat$, $FootIncl$, $Arm1$, $Arm2$. (b) Variável $ComX$. (c) Variáveis $LegOpen$, $Arm0$, $ComY$ e $SideOpen$.

Para que este modelo geométrico leve em consideração o centro de massa junto com a postura de referência do robô, foram adicionadas algumas operações antes do cálculo da cinemática inversa.

Primeiramente, obtém-se a posição do dentro de massa do robô para a configuração de referência $\boldsymbol{B}$ devidamente configurada. Para isso, os seguintes passos são seguidos:

- Subtrai-se da posição vertical do pé (no eixo Z) o valor de $Squat$, Fig. 4.8(a), e acrescenta o valor de $LegOpen$, Fig. 4.8(c), no eixo Y.

$$\begin{aligned} \widetilde{p_y} &= p_y + LegOpen \\ \widetilde{p_z} &= p_z - Squat \end{aligned} \tag{4.12}$$

- Adiciona a inclinação desejada para o pé como valor $FootIncl$, e para o tronco com $Incl$ em torno do eixo Y, Fig. 4.8(a):

$$\begin{aligned} \widetilde{q_y} &= q_y + Incl \\ \widetilde{p_y} &= p_y + FootIncl \end{aligned} \tag{4.13}$$

- Faz a translação de $ComX$, Fig. 4.8(b), e $ComY$, Fig. 4.8(c), nos eixos correspondentes:

$$\begin{aligned} \widetilde{q_x} &= q_x + ComX \\ \widetilde{q_y} &= q_y - ComY \end{aligned} \tag{4.14}$$

- Calcula-se o modelo geométrico inverso para as duas pernas utilizando os novos vetores $\widetilde{\boldsymbol{P}}$ e $\widetilde{\boldsymbol{Q}}$ com os valores acima desejados para as duas pernas, e obtém-se os ângulos para os 12 graus de liberdades correspondentes às pernas.

$$\begin{bmatrix} \tilde{\theta}_0 \\ \tilde{\theta}_1 \\ \tilde{\theta}_2 \\ \tilde{\theta}_3 \\ \tilde{\theta}_4 \\ \tilde{\theta}_5 \end{bmatrix} = CinemáticaInversa(\widetilde{\boldsymbol{P}}, \widetilde{\boldsymbol{Q}}, \boldsymbol{W}) \tag{4.15}$$

- As posições angulares dos braços são obtidas diretamente pelos valores $Arm0$, $Arm1$ e

$Arm2$, e são configuradas simetricamente para ambos os lados.

- Com os 18 graus de liberdades calculados para a postura de referência $\boldsymbol{B}$, é obtido a posição do centro de massa utilizando a função da biblioteca RBDL que tem como entrada o vetor de posições angulares e como saída a posição tridimensional de CM.
- Com base nesse valor de CM, obtém-se os valores de referência para a aplicação do LIPM, em que z_c é a componente do eixo Z do vetor do centro de massa, e Y é a componente S_y.

As equações e programa utilizado para a obtenção da Cinemática Inversa e Cálculo do Centro de massa estão detalhados no Apêndice B. O gerador de trajetórias LIPM está descrito e detalhado no Apêndice C.

4.3 Resultados e discussões

Foram realizados testes de caminhada tanto para o robô real quanto para o simulado. Os parâmetros utilizados para ambos são mostrados na Tabela 4.3.

Tabela 4-2 - Parâmetros utilizados tanto no robô real quanto no robô simulado durante a caminhada.

Parâmetros	Robô simulado	Robô real
T_s	0.27 s	0.10 s
T_d	0.00 s	0.00 s
z_c	0.47 m	0.47 m
v_x	0.10 m/s	0.10 m/s
H_{step}	0.05 m	0.05 m
γ	1	3.4
$Incl$	0.0 graus	15.0 graus
$SideIncl$	0.0 graus	1.8 graus
$FootIncl$	0.0 graus	0.0 graus
$Squat$	0.05 m	0.08 m
$LegOpen$	0.05 m	0.08 m
$ComX$	0.00 m	0.04 m
$ComY$	0.00 m	0.01 m
$Arm0$	0.0 graus	0.0 graus
$Arm1$	0.0 graus	0.0 graus
$Arm2$	45.0 graus	45.0 graus

Observando a Tabela 4.3 percebe-se, primeiramente, que o tempo de simples apoio T_s, tem valores diferentes entre o robô simulado e o real, e esse parâmetro mostrou grande influência para a estabilidade lateral, sendo que quanto maior os valores de T_s, maior o deslocamento lateral gerado pelo LIPM, como pôde ser verificado nos testes da Fig. 4.6, e esse aumento dessa amplitude gerou desestabilizações, como foi observado durante os testes. No caso contrário, com a redução excessiva do valor de T_s, também foi observado desestabilidades laterais, em que um dos pés do robô humanoide não consegue tocar o chão antes do início do movimento de levantar do outro pé.

Para o robô simulado, o período de $T_s = 0.27\ s$ demonstrou ser capaz de manter o balanço do robô durante a caminhada, e para o robô real, observou-se que as folgas mecânicas ampliaram o movimento lateral, e por esse motivo o robô não conseguiu manter o movimento lateral de forma estável, já que tendia a quedas laterais constantemente. Por esse motivo, foi reduzido o tempo de simples apoio para o valor de 0.1 s, e para que o movimento não se tornasse muito rápido à ponto de não conseguir completar a troca de pés no tempo correto, o fator γ foi aumentado gradativamente, até que o robô atingiu uma estabilidade lateral para o movimento lateral para o valor de $\gamma = 3.4$.

As variáveis $ComX$ e $Incl$ para o robô real teve importância na compensação das folgas mecânicas relacionadas ao movimento frontal do robô, já que para os valores $ComX$ e $Incl$ nulos o robô apresentava constantes quedas de costas, e esse valor foi sendo aumentado gradativamente até que o seu centro de massa fosse deslocado para frente para que as quedas fossem reduzidas.

Tanto $SideIncl$ quanto $ComY$, foram importantes para compensar as assimetrias observadas no robô real visualmente, e esses valores foram ajustados até que tanto visualmente, quanto na análise dos torques dos motores durante a movimentação apresentassem comportamento simétrico. Já no robô simulado, essas variáveis não precisaram de ser ajustadas, pois o robô apresentou visivelmente simetria durante a sua caminhada.

O valor de $LegOpen$ mostrou importância na estabilidade lateral do movimento de caminhada, sendo que quanto mais distante uma perna da outra, menor a propensão do robô cair lateralmente, e essa distância teve que ser maior no robô real do que no simulado, principalmente por conta das folgas mecânicas.

Por fim, o valor de $Squat$ tem o efeito de rebaixamento do centro de massa verticalmente, e foi constatado que quanto mais baixo o centro de massa, mais estável é o movimento de caminhada do robô, porém, para o robô real, verificou-se que quanto mais agachado o robô, mais rápido ocorria o aquecimento dos servomotores da perna, e isso é um sinal de torques elevados, o que pode prejudicar os servomotores. Portanto, foi

necessário encontrar valores de $Squat$ intermediários, que apresentassem estabilidade da caminhada e ao mesmo tempo não gerassem torques excessivos nos servomotores. De toda forma, o agachamento necessário para o robô real foi maior que no robô simulado, e isso ocorreu devido às tentativas de reduzir as folgas mecânicas presentes nos motores do calcanhar.

Os testes realizados em simulação são apresentados na Fig. 4.9. O robô demonstrou relativa facilidade de locomoção, sofrendo quedas após pelo menos 30 segundos de caminhada.

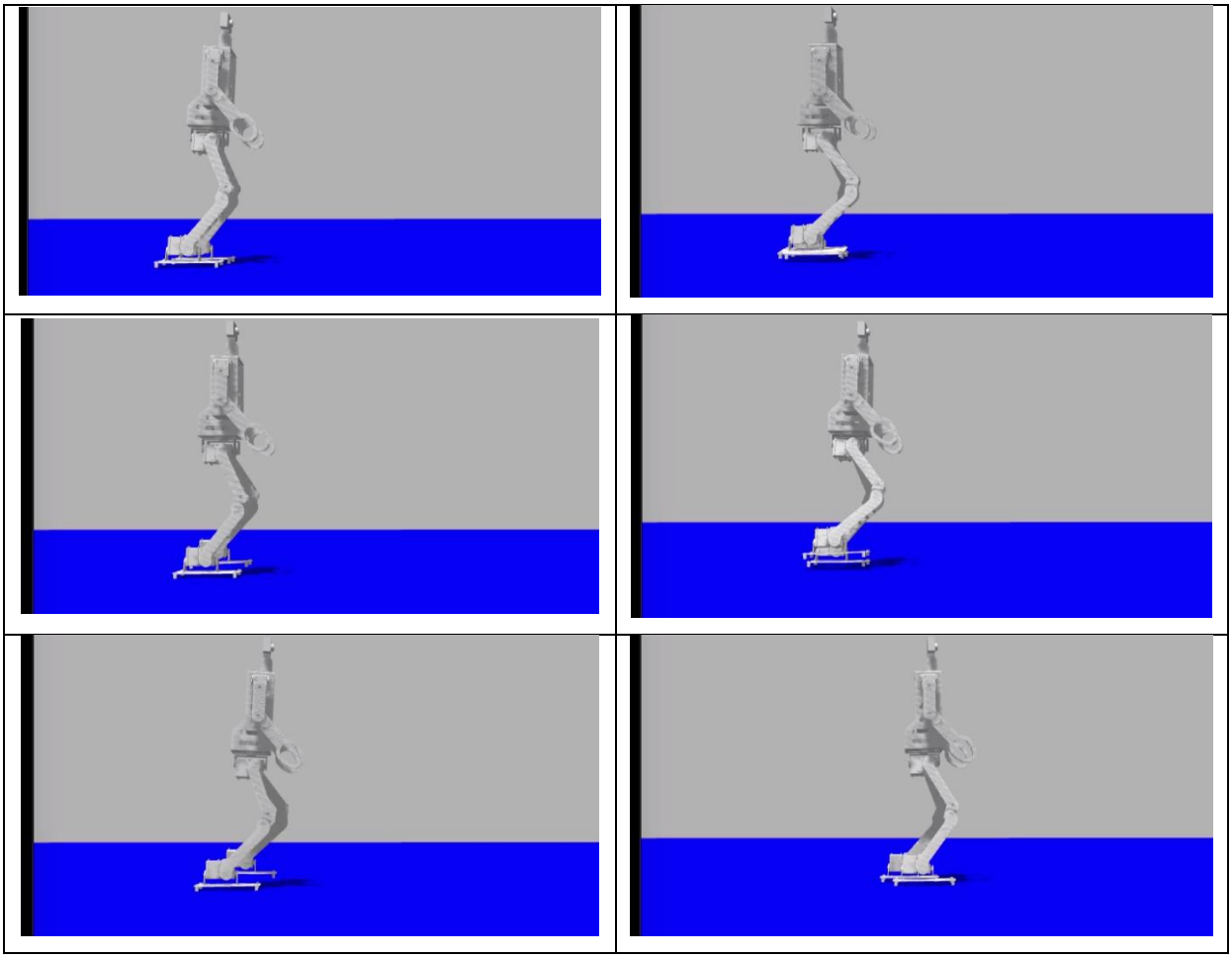


Figura 4.9 – Sequência de 2.4 segundos de caminhada simulados e com as telas capturadas em tempos igualmente espaçados.

O vídeo completo da simulação do robô humanoide pode ser visualizado em: https://drive.google.com/open?id=1wff_bi1lr0-cbb6c06bc11PvbWBy47Q7.

Os testes no robô real, Figura 4.10, mostraram a importância da utilização das variáveis acrescentadas. Observa-se que o vetor $\boldsymbol{B}$ nesse caso apresenta assimetria lateral, com valores de *FootIncl* e *ComY,* e valores relacionados às correções do efeito das folgas,

ou seja, as variáveis *Incl*, *ComX* e γ precisaram de serem configuradas para que o robô conseguisse se locomover mesmo sob influência dos efeitos mecânicos indesejados. O robô apresentou movimentos de caminhada limitados à no máximo 5 passos, o que faz das estratégias de estabilização, que serão apresentadas no Capítulo V, algo essencial para a redução de quedas durante a caminhada do robô.

O vídeo completo do robô humanoide pode ser visualizado em: https://drive.google.com/open?id=15fWZQBt3F24HvJT5ng4uLT4n_QmQtQzO.

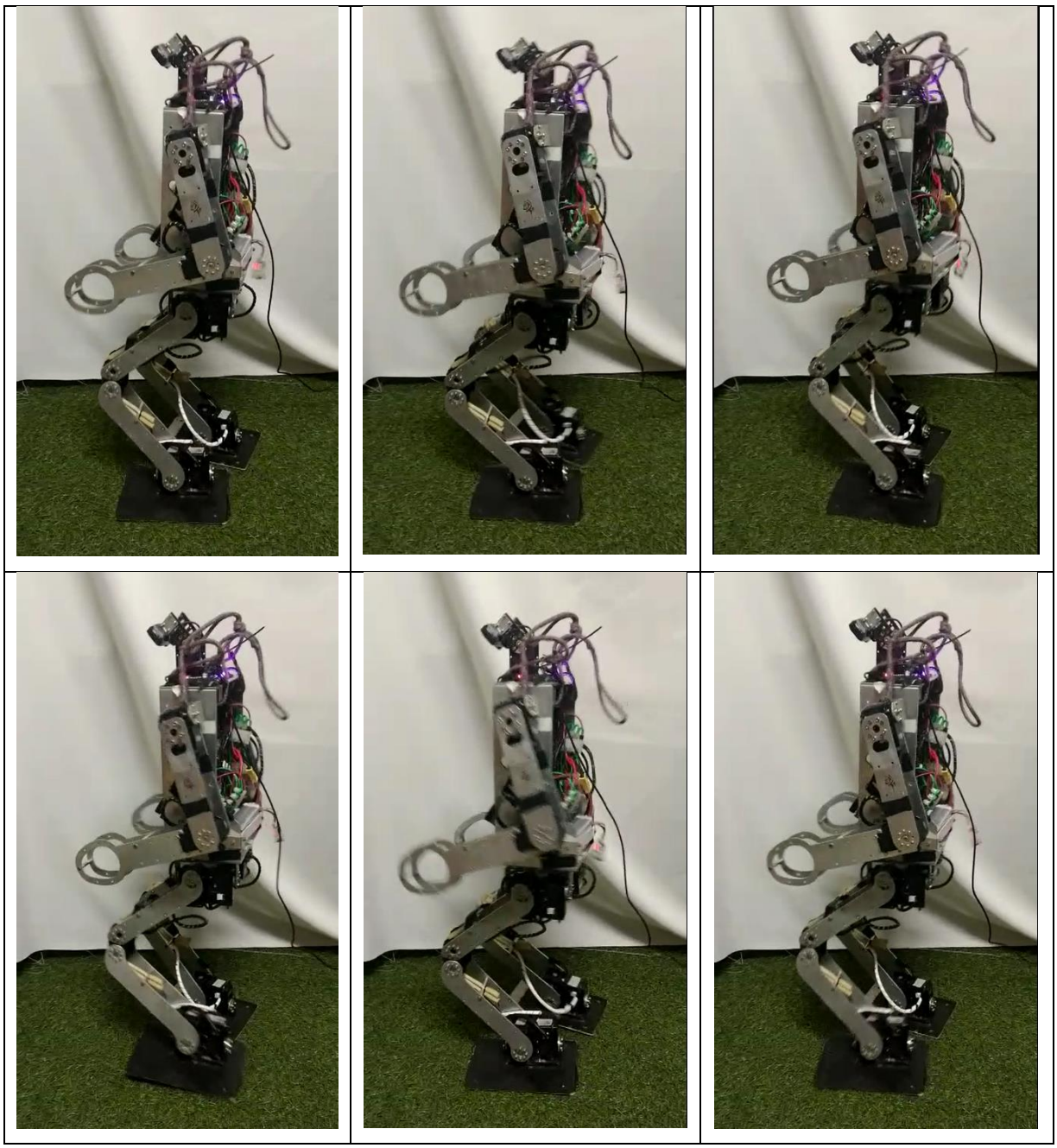


Figura 4.10 – Sequência de 2 passos do teste com o robô real aplicando LIPM com os parâmetros da Tab. 4.3.

4.3.1 Análise de Sensibilidade

Foi realizada uma análise de sensibilidade para o robô simulado com o objetivo de verificar qual a ordem de importância de cada parâmetro de entrada para a distância percorrida pelo robô humanoide. Para isso foi utilizado o Método dos Efeitos Elementares (EE, *Elementary Effects*) com algumas modificações propostas por Campolongo; Cariboni e Saltelli (2007). Trata-se de um método para análise de sensibilidade que permite identificar e classificar quais elementos da entrada de um sistema são mais ou menos importantes ao analisar uma determinada saída de interesse, e isso pode ser obtido para sistemas com custo de execução alta, ou seja, em que cada vez que suas entradas são testadas, o tempo de resposta da saída é relativamente alto. Portanto, esse método é adequado para a presente situação, uma vez que cada execução da caminhada do robô humanoide, leva aproximadamente 1 minuto até que sejam simulados 6 segundos em um computador de processador Intel® Core(TM) i7 2.6 GHz com 8 GB de memória RAM, o que indica o alto custo computacional.

Matematicamente, um sistema a ser analisado pode ser modelado como uma saída Y, que é função de k fatores de entrada:

$$Y = f(\boldsymbol{X}) = f(X_1, X_2, \dots X_k) \tag{4.16}$$

O método original de Morris para o cálculo de medidas de sensibilidade para cada fator de entrada X_i, consiste na construção de diversas trajetórias no espaço das entradas, onde as entradas são modificadas aleatoriamente uma por vez, e cada uma varia de p diferentes valores, também chamados de níveis. Cada trajetória é composta por $(k+1)$ pontos, e é considerado que cada fator de entrada se move um a cada vez. Os deslocamentos de cada variável são dados por $\Delta = \{0\,, 1/(p-1)\,, 2/(p-1), \dots, 1\}$. Com as trajetórias calculadas, é possível definir o efeito elementar de cada entrada através da equação:

$$d_i(\boldsymbol{X}) = \frac{Y(X_1, \dots, X_{i-1}, X_i + \Delta, X_{i+1}, \dots, X_k) - Y(\boldsymbol{X})}{\Delta} \tag{4.17}$$

Ao amostrar r trajetórias, é possível calcular uma medida que ranqueia os fatores de entrada X_i em ordem de importância. A medida que será utilizada nesta dissertação, é a proposta por Campolongo; Cariboni e Saltelli (2007), em que com apenas a média dos valores absolutos de d_i, é possível realizar a classificação desejada. Em outras palavras,

trata-se de um valor μ^* dado por:

$$\mu_i^* = \frac{1}{r}\sum_{j=1}^{r}\left|d_i\left(\boldsymbol{X}^{(j)}\right)\right| \tag{4.18}$$

O valor de μ_i^* é diretamente proporcional ao grau de importância de um determinado fator de entrada X_i. Campolongo; Cariboni e Saltelli (2007) mostram ainda uma abordagem que permite melhorar o método de geração das trajetórias descritas pela Eq. (4.17), em que novas amostragens são feitas para uma exploração mais uniforme do espaço das entradas $\boldsymbol{X}$. Este método foi aplicado para a geração de 1500 trajetórias de entrada para o cálculo de μ_i^*, e a distribuição dos valores de entrada podem ser visualizados na Figura 4.11. A quantidade de níveis utilizados é de $p = 10$, e as variáveis analisadas foram as mesmas descritas pela Tabela 4.2, porém, com valores fixos para $z_C = 0.47\ m$, e para $v_x = 0.16\ m/s$.

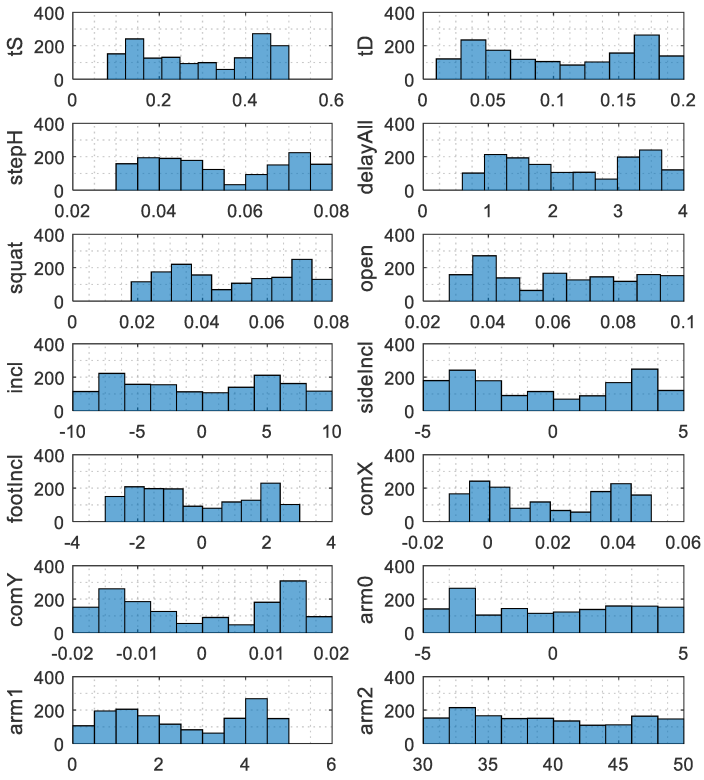


Figura 4.11 – Histogramas com a distribuição das 1500 trajetórias das variáveis de entrada ao longo dos $p = 10$ níveis utilizados para a análise de sensibilidade.

Os histogramas da Figura 4.11 mostram os intervalos analisados para cada variável de entrada no seu eixo das abscissas, sendo que estes valores foram baseados nos valores utilizados na Tabela 4.2. Percebe-se visualmente que, no geral, as distribuições mostradas

tendem à uma distribuição normal dos dados, e isso é importante para o ranqueamento dos fatores X_i (CAMPOLONGO; CARIBONI; SALTELLI, 2007).

A variável de saída Y analisada, trata-se do deslocamento total do robô antes de sofrer queda na simulação para um determinado conjunto de entrada $\boldsymbol{X}$. Essa característica foi escolhida por indicar se um determinado conjunto de parâmetros é bom ou ruim para a caminhada e estabilidade do robô, sendo que quando o robô caminha por longas distâncias, é esperado que os valores de $\boldsymbol{X}$ proporcionaram uma estabilidade adequada para o balanço do robô humanoide. Dessa forma, com a análise proposta, é possível verificar quais as variáveis que devem ser mais ou menos levadas em consideração durante o projeto da caminhada. A Tabela 4.3 mostra a classificação obtida.

Tabela 4-3 – Resultados da análise de sensibilidade pelo método dos Efeitos Elementares (EE).

Parâmetro	$\boldsymbol{\mu_i^*}$
γ	0.528
$FootIncl$	0.500
$Incl$	0.495
T_d	0.475
$ComX$	0.457
T_s	0.444
$Arm0$	0.408
H_{step}	0.404
$ComY$	0.390
$Squat$	0.389
$Arm1$	0.384
$Arm2$	0.357
$LegOpen$	0.353
$SideIncl$	0.332

Segundo os resultados obtidos, pode-se dizer que dentre os parâmetros que mais influenciam a estabilidade do robô está o fator de tempo γ, que tem um papel importante no ajuste do tempo T_s. Em seguida estão dois valores relacionados à inclinação frontal do robô, $FootIncl$ e $Incl$, sendo que a maior parte dos momentos de queda do robô ocorrem com este caindo ou de frente ou de costas, e, portanto, o ajuste dessas variáveis auxilia na estabilidade geral do movimento. Dentre os menos importantes estão dois parâmetros

relacionados à movimentos laterais do robô, os parâmetros $LefOpen$ e $SideIncl$, o que demonstra que assimetrias neste eixo não chegam a causar quedas tão frequentes como ocorre no eixo frontal de movimento. É importante apontar que este teste pode ser apenas um indicador para investigações mais profundas de análise de sensibilidade, que pode ser feito inclusive para variáveis de saída mais elaboradas e que indiquem estabilidade levando em consideração outros fatores, como: o balanço do movimento, a posição do centro de massa, dentre outros.

CAPÍTULO V

CONTROLE DE IMPEDÂNCIA

A medida que o robô caminha, alguns distúrbios externos, como a modificação da rigidez do terreno, pequenos obstáculos, e a vibração causada pelo impacto do robô com o piso, geram efeitos que prejudicam a estabilidade do movimento. Uma das propostas adotadas para reduzir esses efeitos indesejados, é o controle de impedância, que, basicamente, trata-se dos conceitos de rigidez e amortecimento para cada articulação do robô e auxilia o sistema na rejeição de distúrbios sofridos.

Este controle é citado na literatura, geralmente em conjunto com o controle de admitância, como métodos de controle para mecanismos robóticos em que o ambiente externo deve ser levado em consideração (HOGAN, 1985).

A necessidade de aplicação destas abordagens surge quando, por exemplo, um braço robótico deve realizar tarefas em que o ambiente não está acuradamente modelado, e, portanto, o robô pode encostar em paredes, ou mesmo em seres humanos. Porém, o robô não pode danificar-se ou mesmo ferir pessoas. O ideal é que este braço robótico, caso interaja com o ambiente, possa aplicar uma força conhecida sobre os obstáculos (restrições). Conforme citado em Siciliano *et al.* (2008), as técnicas clássicas de controle que levam em consideração puramente a posição e velocidades do robô tendem a falhar em situações como esta.

O controle por impedância pode ser descrito, em sua forma mais básica para apenas um grau de liberdade, pela seguinte equação:

$$\mathrm{F} = \mathrm{K}[x_0 - x] + \mathrm{B}[\dot{x}_0 - \dot{x}] \tag{5.1}$$

Onde:

x : posição atual do efetuador

x_0 : posição desejada para o efetuador (referência)

$\dot{x}$: velocidade atual do efetuador

$\dot{x}_0$: velocidade desejada para o efetuador (referência)

K : constante de rigidez do controlador

B : constante de amortecimento do controlador

F : força necessária para o efetuador alcançar a posição e velocidade desejados

A Eq. (5.1) pode ser comparada à equação obtida de um sistema massa-mola amortecedor, em que há uma constante de rigidez e de amortecimento, ou seja, é como se este tipo de controle pudesse “simular” um comportamento dinâmico de um sistema deste tipo. Um dos benefícios da aplicação deste controle para as articulações robóticas, é a possibilidade de modificar a rigidez e o amortecimento do sistema conforme a necessidade. No caso desta dissertação, estes valores serão modificados conforme os distúrbios presentes durante a locomoção.

Sabe-se que um sistema com amortecimento é capaz de absorver/dissipar mais ou menos energia quando perturbado, e isso depende dos seus valores de amortecimento e rigidez. Quanto maior o valor de rigidez e menor o valor de amortecimento, maior será a oscilação do sistema ao ser perturbado, e vice-versa. É importante salientar, que no caso de um robô bípede, se cada articulação apresentar oscilações constantemente, ou seja, se o robô começar a vibrar, a probabilidade do robô cair aumentará. Além disso, para pisos com alta rigidez (cerâmica, por exemplo), a chance do robô apresentar vibrações no sistema também é maior, e isso pode ser corrigido caso a rigidez da perna esteja com um valor necessário para absorver os impactos advindos da interação do pé com o solo.

A Eq. (5.1) pode ser aplicada para um motor, relacionando o deslocamento e velocidade angular (θ e ω) com o torque (τ), sendo os valores desejados para cada grandeza dados pelas variáveis θ_0 e ω_0. Esta relação tem forma semelhante à anterior, e é descrita como:

$$\tau = \mathrm{K}[\theta_0 - \theta] + \mathrm{B}[\omega_0 - \omega] \tag{5.2}$$

É importante observar que a Eq. (5.2) representa um controlador PD (proporcional-derivativo), ou seja, o controle de impedância nada mais é do que um controle de posição, porém, os valores de K e B são escolhidos visando a interação com o ambiente, e K e B passam a ser vistos como as constantes K_p e K_d do controlador PD respectivamente.

Os servomotores digitais que são utilizados neste projeto são do modelo MX-106 da fabricante Dynamixel. Estes servos possuem controle interno de posição PID já implementado (ROBOTIS, [S.d.]). Pode-se modificar apenas as variáveis responsáveis pelos ganhos K_p e K_d do motor, com o valor nulo para o ganho K_i.

Pequenos valores de K_p e K_d, que podem ser desejados para a adaptação ao piso, fazem com que o motor não consiga seguir a trajetória desejada corretamente, apresentando um atraso e diminuição da amplitude do movimento. Esse efeito é observado por Missura (2016), e ele observa ainda que valores baixos de K_p e K_d fazem com que o motor se comporte como um filtro passa baixa, ou seja, caso o movimento desejado para um determinado motor tenha uma frequência relativamente alta para os parâmetros de controle K_p e K_d estabelecidos, o motor realiza um movimento com uma frequência menor e uma amplitude menor que a original e com um atraso de fase em relação ao movimento desejado. A Figura 5.1 mostra este efeito graficamente, onde o atraso é perceptível entre os valores desejados e medidos, uma vez que as posições medidas se manifestam após um determinado período após a posição desejada, e com uma amplitude do movimento menor, que fica evidente no movimento oscilatório entre os segundos 6 e 8.

Figura 5.1 - Efeito de um filtro passa-baixa para um valor reduzido de K_p.

Uma das formas de lidar com este efeito indesejado é através de uma compensação *feedforward* do movimento desejado de entrada, ou seja, com o conhecimento do torque que será solicitado para o movimento através da dinâmica inversa do robô e com um modelo acurado do motor, é possível compensar a trajetória desejada na entrada do controlador afim de que o motor alcance as posições desejadas. Este procedimento é apresentado em Schwarz e Behnke (2013). A Figura 5.2 mostra um esquema deste método.

A principal vantagem da aplicação do método de controle de impedância em conjunto com a compensação *feedforward* é de reduzir as vibrações ocasionadas no robô durante sua caminhada, mantendo ao mesmo tempo uma boa acurácia para seguir as trajetórias solicitadas. Além dessa vantagem, é esperado que, uma vez que os ganhos K_p e K_d são menores do que normalmente se utiliza para o controle de posição de servomotores, o

motor consuma menos corrente que o normal, e, portanto, que aqueça menos.

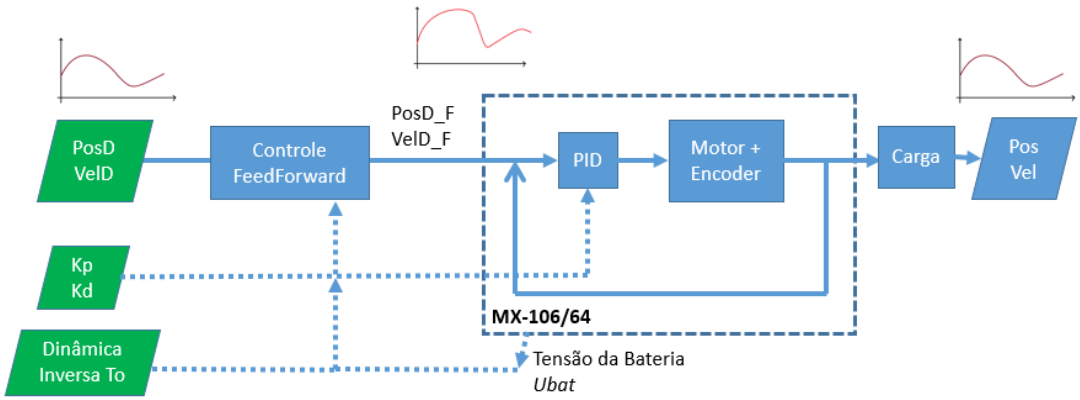


Figura 5.2 – Esquema do controle compensador aplicado em Schwarz e Behnke (2013)

para baixos valores de K_p e K_d do controlador interno do motor MX-106. Resumidamente, a posição e velocidade desejada ($PosD$ e $VelD$) são modificadas pelo *Controle FeedForward* e se transforma em $PosD\_F$ e $VelD\_F$, e dessa forma o motor consegue fazer com que a posição e velocidade final (Pos e Vel) fiquem próximas de $PosD$ e $VelD$.

Este controlador foi implementado em apenas um motor para verificar sua funcionalidade e eficácia na prática. O sistema de um grau de liberdade consiste, em um motor do modelo *Dynamixel* MX-106, que possui como torque máximo 106 Kg.cm. Foi preso em seu eixo uma haste de 3.5 cm, e na sua extremidade foram parafusados objetos de massas conhecidas. O sistema, assim como as medidas físicas relevantes são detalhados na Fig. 5.3. Uma peça de alumínio foi deixada em uma posição que interrompe o movimento do sistema em determinados momentos da trajetória desejada. Essa interrupção da trajetória foi proposital, e com o objetivo de avaliar a corrente de pico do sistema para diferentes valores de rigidez K_p. É esperado que com valores menores que o valor de fábrica o pico da corrente seja reduzido.

O método de Schwarz e Behnke (2013) consiste na aplicação de uma trajetória conhecida para que seja seguida pelo sistema, e durante a execução do movimento são obtidos os valores de posição na saída do eixo do motor. Após uma execução completa da trajetória, é calculado o erro absoluto, ponto a ponto entre o valor desejado e o obtido na saída do sistema. Esse erro é utilizado para o cálculo de uma nova trajetória de entrada, através da seguinte equação:

$$U^{(0)}(t) = 0, \tag{5.3}$$

$$U^{(k+1)}(t) = U^{(k)}(t) + \lambda(e^{(k)}(t)) \tag{5.4}$$

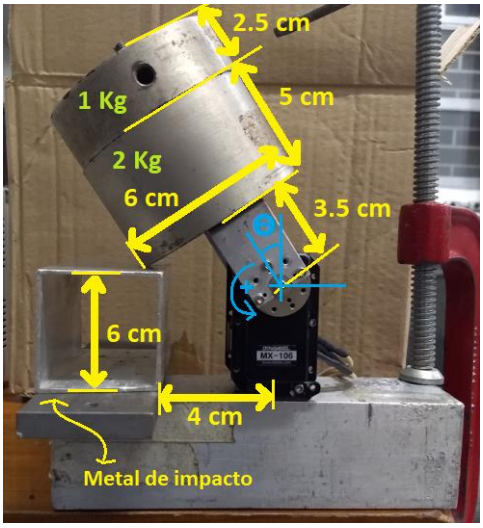


Figura 5.3 – Sistema elaborado para o teste do método de controle de impedância com compensação *feedforward*.

A variável $U(t)$ é um valor escalar que compõe os pontos da curva que será adicionada à entrada desejada do sistema na iteração k no tempo t. Portanto, essa variável é atualizada a cada iteração $k+1$, e isso ocorre através da soma do valor da iteração passada k com o produto entre coeficiente de aprendizagem λ com o erro $e^{(k)}$ da última iteração. Esse erro é calculado com a diferença no tempo dos valores desejados para a saída do sistema, e os valores obtidos na saída. A curva utilizada para ser repetida a cada iteração é mostrada na Fig. 5.4 e foi obtida através da soma de equações senoidais de frequências e amplitudes diferentes conforme apresentada na Eq. 5.5. Essa escolha se baseou no fato de que essa característica oscilatória de movimento e com amplitude variada é comum durante o movimento de caminhada (SCHWARZ; BEHNKE, 2013).

$$q_d(t) = \sin\left(\frac{3t^{1,2}}{2}\right) - \left(\frac{t}{20} - \frac{1}{10}\right)^3 + \cos\left(\frac{27t^{1,3}}{20}\right)^3 + 1 \tag{5.5}$$

A atualização dessa nova trajetória de entrada e a repetição desse processo, consiste em um método de aprendizagem conhecido como ILC (*Iterative Learning Control*) (MOORE, 1993), em que é demonstrado que para sistemas de uma entrada e uma saída, o erro máximo entre o desejado e a saída do sistema tende a diminuir gradativamente à medida que o número de iterações aumenta. Dessa forma, é possível obter qual a entrada que o motor deveria ter para reproduzir a trajetória desejada.

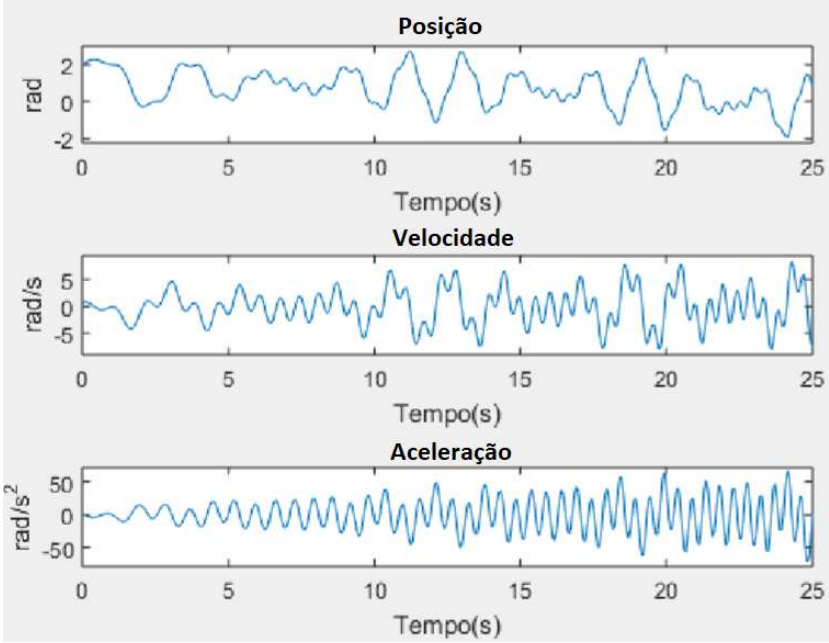


Figura 5.4 – Posição, velocidade e aceleração da trajetória escolhida.

Tanto o envio da trajetória seguida, a configuração do PID, quanto a leitura das grandezas dos servomotores são obtidos do próprio servomotor através da comunicação com o seu circuito interno. Para avaliar a melhoria do sistema a cada execução da trajetória, foi utilizado o erro RMS para a curva de erros ao longo do tempo de $e^{(k)}(t)$.

Tendo em mãos a trajetória corrigida, e a desejada, Schwarz e Behnke (2013) apresentaram o modelo matemático que faz a transformação entre ambas. Esse modelo matemático é obtido através da modelagem do servomotor que contempla: o atrito estático e o de Coulomb, o torque necessário para a execução do movimento desejado, o valor de tensão da bateria, e o modelo interno do controle do motor considerando apenas um controle proporcional. As Eqs. (5.5)-(5.7) descrevem este modelo completo de forma compactada com as variáveis α_i que devem ser estimadas durante o processo.

$$U(t) = \left(\alpha_0\tau_0(t) + \alpha_1\dot{q}_d(t) + \alpha_2\left(1-\beta(t)\right) + \alpha_3\beta(t)\right) \tag{5.6}$$

$$\beta(t) = exp\left(-\left|\frac{\dot{q_d}(t)}{\dot{q}^{(s)}}\right|^{\delta}\right) \tag{5.7}$$

$$q_f(t) = \frac{U(t)}{k_P U_{bat}(t)} + q_d(t) \tag{5.8}$$

Sendo que, q_d e $\dot{q_d}$, são as posições e acelerações desejados mostrados na Fig. 5.4, q_f, é a posição corrigida à ser enviada para o motor, U_{bat}, é a tensão da bateria no momento do envio da posição, $\dot{q}^{(s)}$ é um parâmetro para a velocidade de transição, e δ

depende da superfície do material de atrito. A variável $\tau_0(t)$ representa o torque necessário para o mecanismo poder executar o movimento desejado, e foi calculado com um modelo do sistema através do método de Newton-Euler, mostrado na Eq. 5.9, em que a massa m é igual a massa da barra junto com o massa da ponta, e o comprimento L corresponde ao comprimento da barra.

$$\tau_0(t) = mgcos\big(q_d(t)\big) - mL^2\ddot{q_d}(t) \tag{5.9}$$

Os testes realizados para o cálculo das variáveis α_i do polinômio $U(t)$, foram feitos sem a peça de alumínio utilizada para a colisão, já que esse seria um fator não modelado para o calculo da dinâmica inversa. Foram realizadas 30 iterações para que o erro RMS não apresentasse redução significativa. Em seguida, foi utilizado o método dos mínimos quadrados (ASTROM; WITTENMARK, 1995) para a regressão linear do polinômio da Eq. 5.5 utilizando todos os pontos da última curva $U(t)$ obtida. Os testes foram feitos para um valor de rigidez 37.5 % do valor de fábrica para três valores de massa distintos: 1 kg, 2kg e 3kg. Estes valores foram escolhidos por abranger situações de torque semelhantes com quando o robô se encontra com a perna levantada ou durante a caminhada. Foram realizados 5 testes para cada uma das três configurações, para que os valores de α_i obtidos pudessem ser comparados e avaliados dentro de um intervalo de confiança. As constantes utilizadas para o cálculo da Eq. (5.6) e os resultados obtidos pra α_i são mostrados nas Tabs. 5.1 e 5.2.

Tabela 5-1 – Valores utilizados para as constantes, baseados nos valores utilizados em Schwarz e Behnke (2013).

$\dot{\boldsymbol{q}}^{(s)}$	$\boldsymbol{\delta}$	$\boldsymbol{\lambda}$
0.1 rad/s	1.0	0.5

Tabela 5-2 – Valores obtidos para as variáveis livres no polinômio de $U(t)$. Os valores de intervalos de confiança para as médias obtidas são para uma confiança de 95 %.

Massa	$\boldsymbol{\alpha_0}$		$\boldsymbol{\alpha_1}$		$\boldsymbol{\alpha_2}$		$\boldsymbol{\alpha_3}$	
1 kg	0.1848	±0,0036	0,2328	±0,0016	0,0433	±0,0019	-0,0164	±0,0025
2 kg	0,3945	±0,0094	0,2673	±0,0029	0,0245	±0,0016	-0,0129	±0,0009
3 kg	1,3177	±0,0652	1,4082	±0,0921	-0,3309	±0,0273	0,0591	±0,0078

Através da Tabela 5.2, percebe-se que os valores de α_i são claramente distintos entre as massas analisadas, principalmente para de valor 3 kg, e isso pode ser explicado devido ao aumento de dificuldade do motor em manter a trajetória proposta em sua saída com o aumento da massa em seu eixo. Esse aumento de esforço pode fazer com que o motor exija do sistema uma corrente maior que ele está projetado, e o sistema pode entrar em saturação, o que introduz efeitos não lineares não modelados pelas equações no trabalho de Schwarz e Behnke (2013), e, portanto, isso pode prejudicar a determinação das variáveis α_i. Outra característica que evidencia a influência da massa na qualidade dos valores de α_i obtidos, é o relativo aumento do intervalo de confiança conforme a massa aumenta, e isso pode ser explicado pelos mesmos argumentos anteriores.

Para a determinação dos valores de α_i que serão efetivamente utilizados para os demais testes de impacto, foram realizados testes de controle de posição com compensação *feedforward* para as três configurações de massa no sistema combinados com os três conjuntos de α_i obtidos. A trajetória utilizada foi gerada pela Eq. (5.5), porém adiantada no tempo de 30 segundos, sendo que esse deslocamento temporal tem por objetivo obter uma curva diferente àquela utilizada para o treinamento, e, portanto, verificar a generalização do método. As Figuras 5.5, 5.6 e 5.7 apresentam os resultados do controle de posição com a aplicação de *feedforward* para 30 repetições do experimento. A curva sólida preta é a posição desejada, e a linha tracejada vermelha representa a média das 30 medidas, e o seu desvio padrão multiplicado por 10 é representado em vermelho sombreado. Essa amplificação do desvio padrão tem como objetivo evidenciar a diferença dos desvios ao longo dos experimentos. A Figura 5.5 mostra que os valores de α_i(1kg) e α_i(2kg) permitiram que o controle de posição apresentasse comportamentos semelhantes, e com valores de erros aparentemente baixos em relação ao intervalo de posição abrangido no teste. Além disso, para essas duas situações, percebeu-se que o valor do desvio padrão é quase imperceptível na figura, mesmo com a sua ampliação. O mesmo comportamento não ocorre para o os valores de α_i(3kg), onde os erros de posição e o desvio padrão são claramente maiores. Os mesmos comportamentos são percebidos para os testes com as massas 2 kg e 3 kg, Figuras 5.6 e 5.7, sendo que para a massa de 3 kg com o valor de α_i(3kg) é o mais afetado, e apresentou os maiores desvios de erro de posição. Isso mostra que como era esperado, a qualidade dos valores estimados para α_i, tende a reduzir com o aumento dos esforços na saída do servomotor.

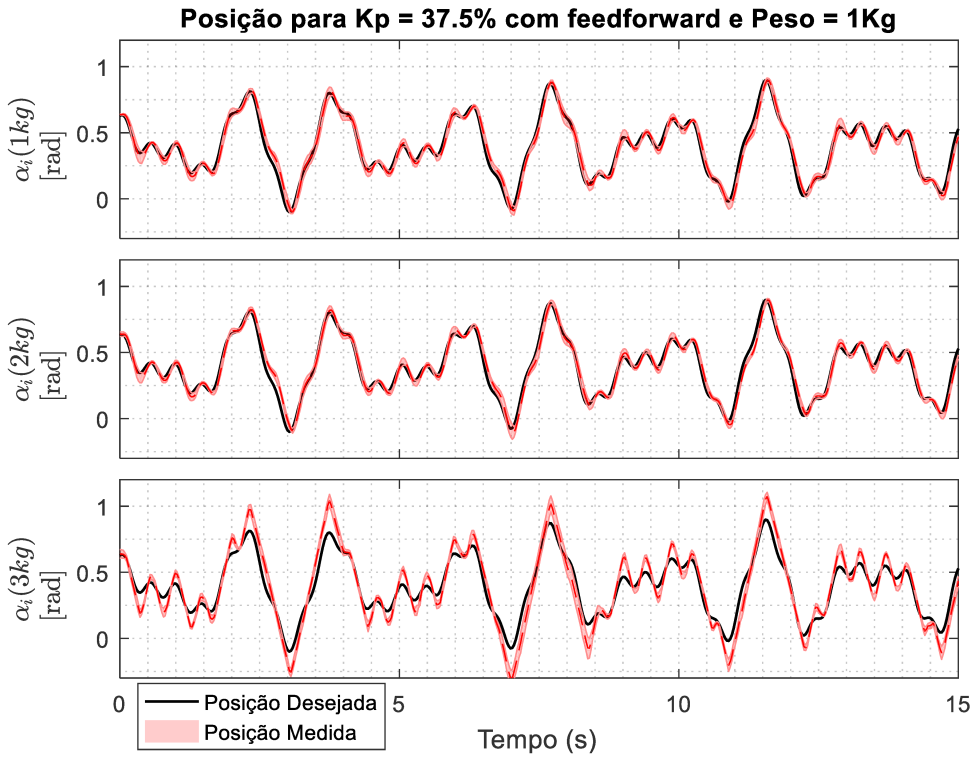


Figura 5.5 – Controle de posição de trajetória para a massa de 1 kg e K_p reduzido com controle *feedforward* para os três diferentes conjuntos de α_i encontrados na Tabela 5.2.

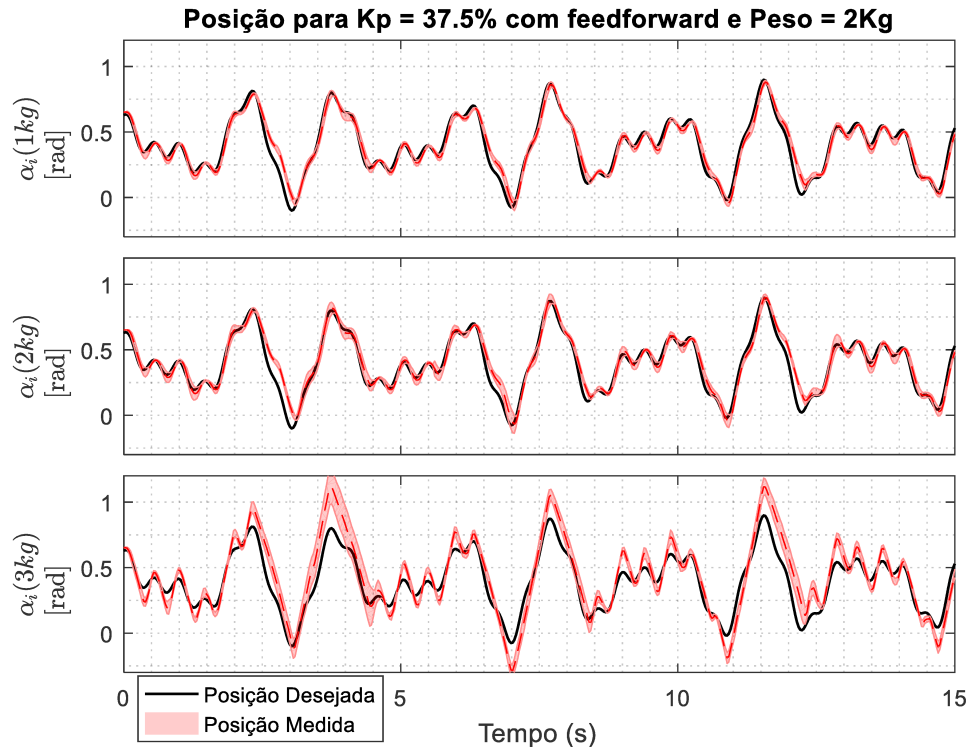


Figura 5.6 - Controle de posição de trajetória para a massa de 2 kg e K_p reduzido com controle *feedforward* para os três diferentes conjuntos de α_i encontrados na Tabela 5.2.

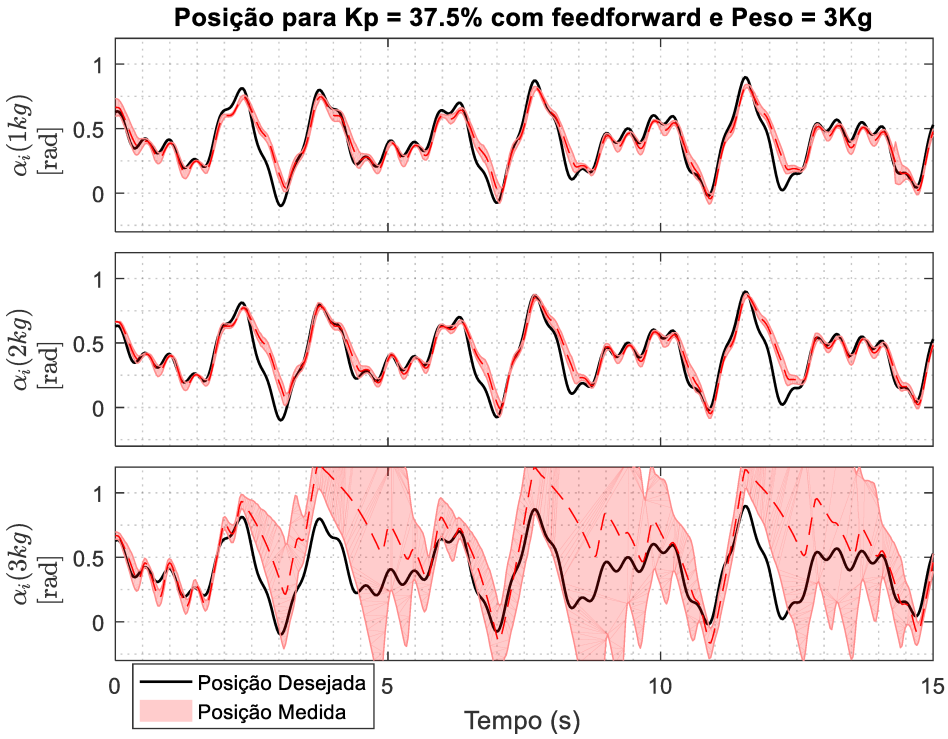


Figura 5.7- Controle de posição de trajetória para a massa de 3 kg e K_p reduzido com controle *feedforward* para os três diferentes conjuntos de α_i encontrados na Tabela 5.2.

A Tabela 5.3 apresenta os resultados numéricos dos experimentos para um intervalo de confiança de 95 %. Os valores obtidos confirmam os efeitos dos diferentes valores de α_i percebidos nas Figuras 5.5, 5.6 e 5.7. Além disso, ao considerar o intervalo de confiança adotado, é possível afirmar que nas três configurações de massa, o valor de α_i(1kg) é o que apresenta os menores erros máximos. Portanto, os valores de α_i(1kg) serão os utilizados para os testes de impacto que serão apresentados a seguir.

Tabela 5-3 – Resultados numéricos para os erros máximos para os experimentos realizados. O intervalo de confiança é de 95 %.

Massa	$\boldsymbol{\alpha_i(1\ kg)}$		$\boldsymbol{\alpha_i(2\ kg)}$		$\boldsymbol{\alpha_i(3\ kg)}$	
	Erro Máximo [rad]		**Erro Máximo [rad]**		**Erro Máximo [rad]**	
1 kg	0.1026	±0.0010	0.1185	±0.0007	0.2412	±0.0018
2 kg	0.1656	±0.0012	0.1913	±0.0009	0.3240	±0.0052
3 kg	0.2813	±0.0023	0.3476	±0.0020	0.8180	±0.0193

Após a determinação do polinômio $U(t)$ com os valores de α_i(1kg), foi realizado o teste com o impacto na peça de alumínio, tanto para uma constante de rigidez de fábrica, quanto para a com redução de 37.5 % com compensação *feedforward* calculada pela Eq. (5.7). Os testes de impacto foram realizados nas mesmas condições por 30 vezes e os dados são exibidos nas Figuras 5.10 à 5.15, onde são mostrados graficamente os resultados no tempo. A curva de corrente elétrica, dada pela linha azul pontilhada, corresponde à média das 30 amostras e tem sobreposta, em sombreado azul, a região correspondente ao nível de confiança de 95 % dos resultados. Percebe-se que existem quatro principais momentos de impacto durante o movimento, próximos aos segundos: 2.3, 3.8, 7.8, e 11.5. Nas regiões de impacto, o controle de posição não consegue fazer com que a massa utilizada empurre a peça de alumínio, e isso gera o pico de corrente visualizado nas figuras, sendo que é claramente visível que para as situações que se utilizou os valores de rigidez de fábrica, a corrente de pico é consideravelmente superior do que para as situações de K_p reduzido e com controle *feedforward.* Além disso, a semelhança na qualidade da trajetória da posição entre ambas situações de rigidez é equivalente, mostrando que o método analisado para a redução da rigidez do movimento é benéfico tanto para a redução das correntes de pico durante impactos, quanto para a redução do efeito passa-baixa provocado pelo baixo valor de K_p.

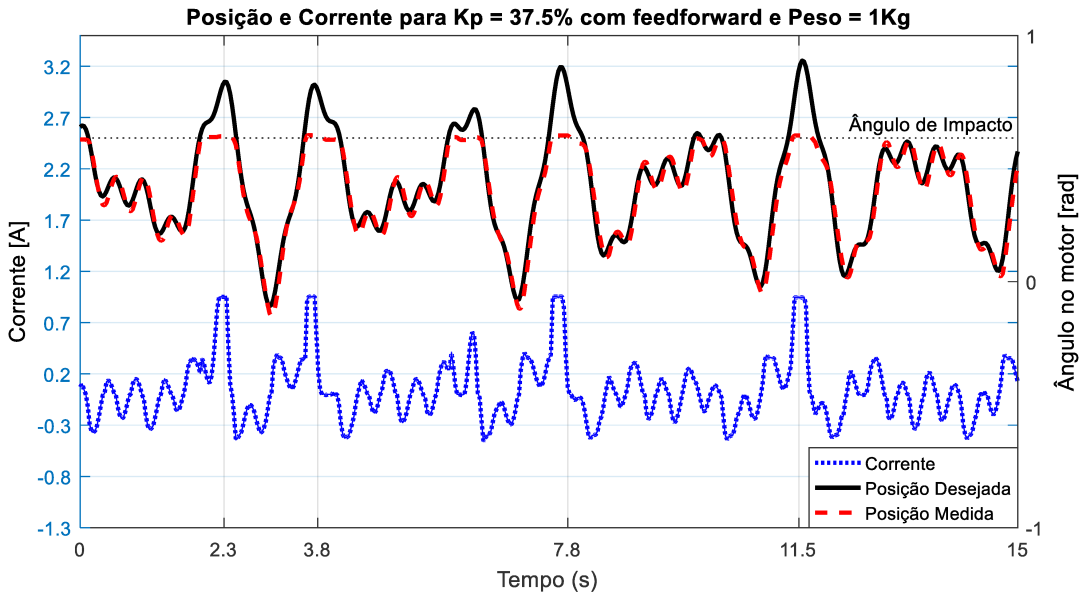


Figura 5.8 - Teste de impacto para uma constante de rigidez de 37.5 % o valor de fábrica com controle *feedforward*.

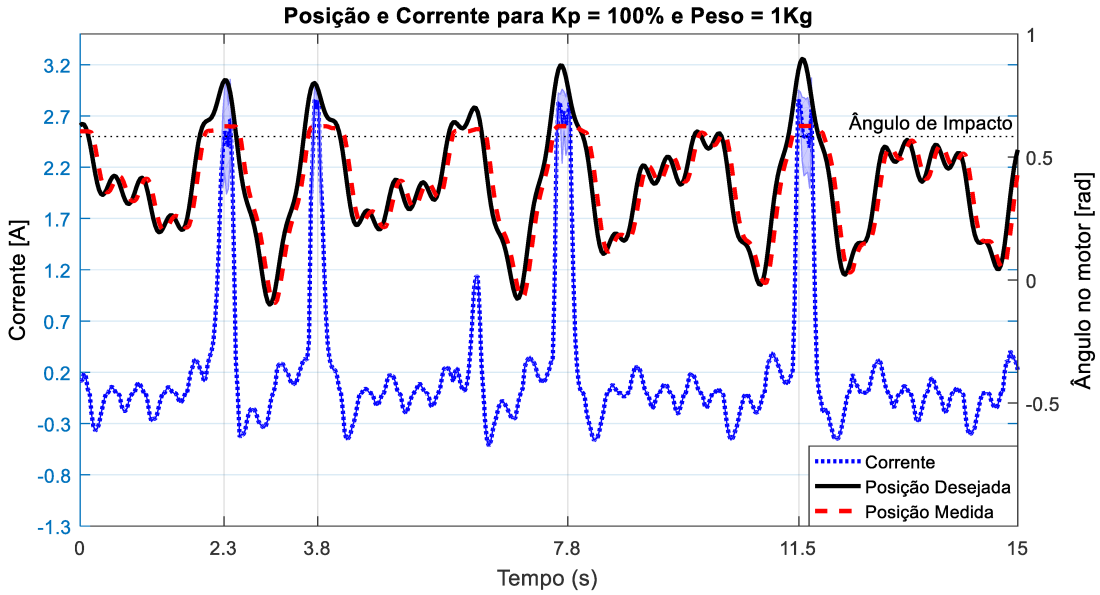


Figura 5.9 - Teste de impacto para uma constante de rigidez de 100 % o valor de fábrica.

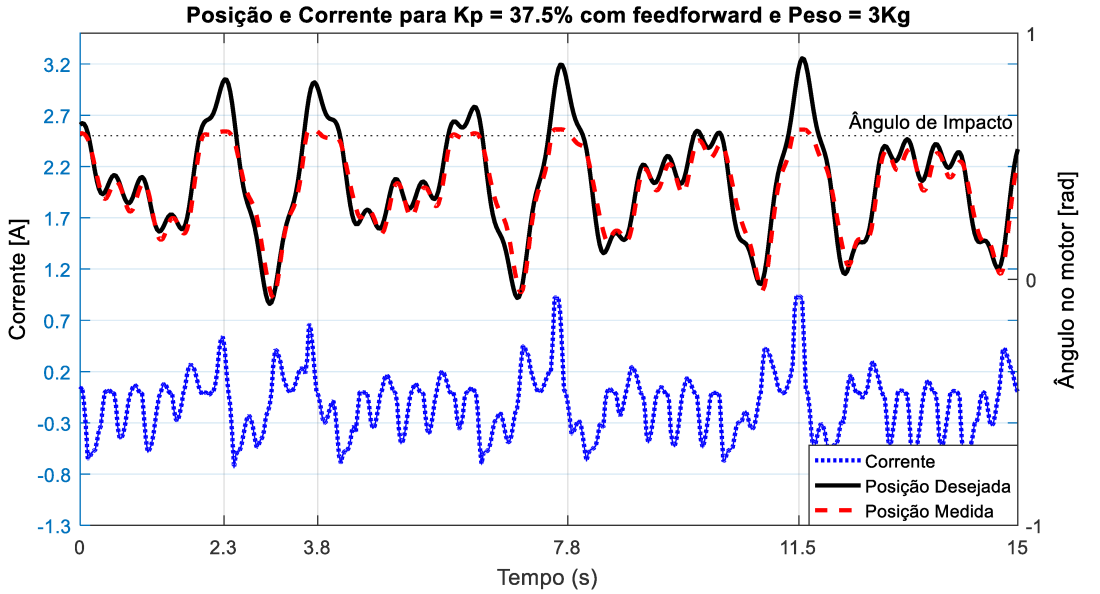


Figura 5.10 - Teste de impacto para uma constante de rigidez de 37.5 % o valor de fábrica com controle *feedforward*.

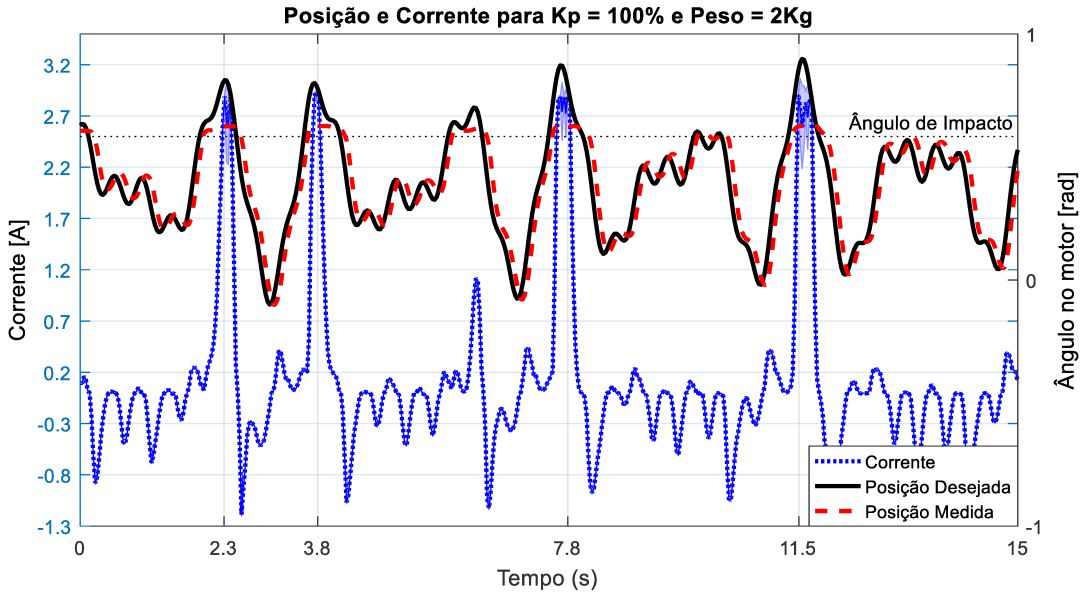


Figura 5.11 - Teste de impacto para uma constante de rigidez de 100 % o valor de fábrica.

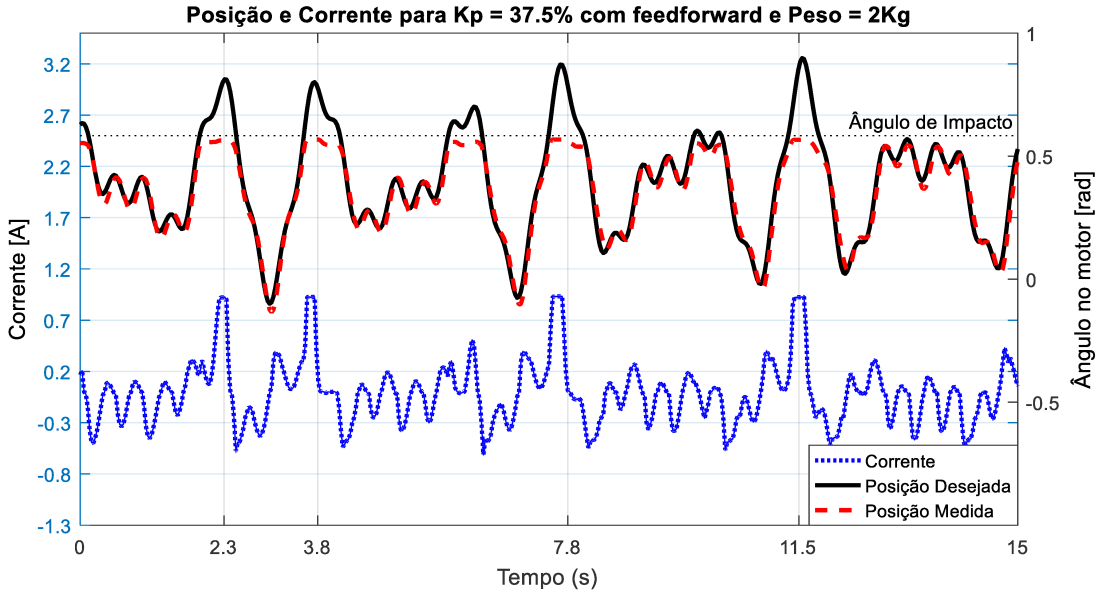


Figura 5.12 - Teste de impacto para uma constante de rigidez de 37.5 % o valor de fábrica com controle *feedforward*.

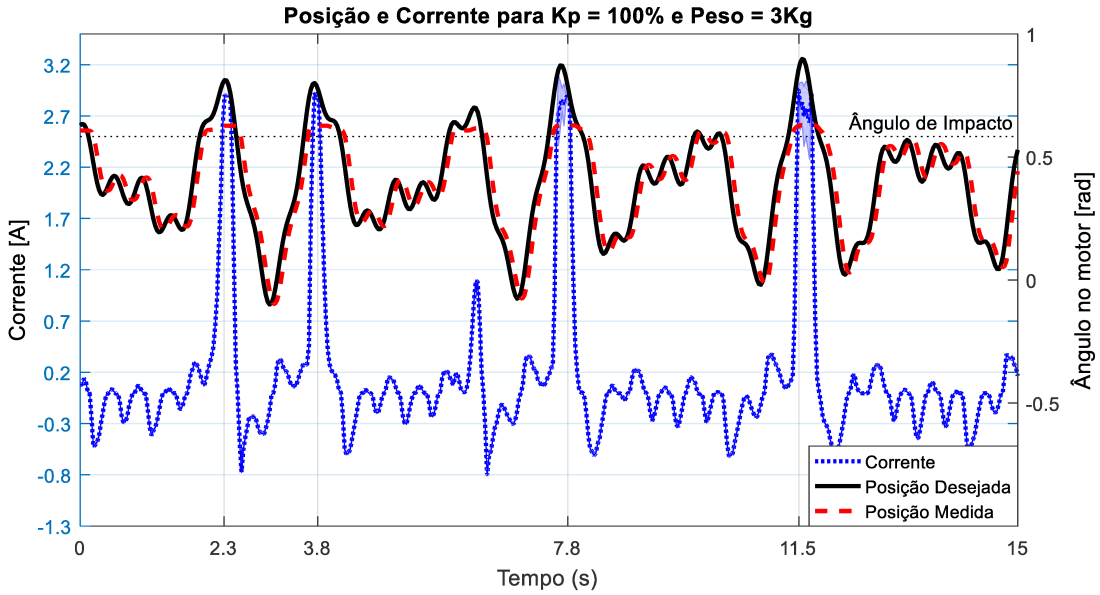


Figura 5.13 - Teste de impacto para uma constante de rigidez de 100 % o valor de fábrica.

As Tabela 5.4 e 5.5 resumem os principais resultados relacionados aos erros de posição máximo, os picos de corrente e a carga elétrica total consumida, obtida da integração da corrente no tempo.

Tabela 5-4 – Resultados numéricos para as correntes de picos dos experimentos de impacto com K_p = 37.5 % com controle *feedforward*, e com K_p = 100 % sem controle *feedforward*. O nível de confiança para os resultados é de 95 %.

Massa	K_p	**Pico 1 (~2.3 s)**	**Pico 2 (~3.8 s)**	**Pico 3 (~7.8 s)**	**Pico 4 (~11.5 s)**
1 kg	**37.5 %**	0.95 ± 0.01 A	0.95 ± 0.01 A	0.95 ± 0.01 A	0.95 ± 0.01 A
	100 %	2.67 ± 0.39 A	2.86 ± 0.12 A	2.68 ± 0.29 A	2.77 ± 0.30 A
2 kg	**37.5 %**	0.92 ± 0.01 A	0.92 ± 0.00 A	0.93 ± 0.01 A	0.93 ± 0.00 A
	100 %	2.64 ± 0.40 A	2.90 ± 0.04 A	2.74 ± 0.29 A	2.75 ± 0.31 A
3 kg	**37.5 %**	0.54 ± 0.00 A	0.66 ± 0.00 A	0.92 ± 0.01 A	0.94 ± 0.00 A
	100 %	2.89 ± 0.01 A	2.92 ± 0.01 A	2.76 ± 0.34 A	2.77 ± 0.27 A

Tabela 5-5 – Resultados numéricos relacionados aos erros máximos de posição (desconsiderando as regiões acima do ângulo de impacto) e de carga elétrica consumida. O nível de confiança para os resultados é de 95 %.

Massa	K_p	**Erro máx. [rad]**	**Carga Consumida [Coulombs]**
1 kg	**37.5 %**	0.0611 ± 0.0013	3.2437 ±0.0106
	100 %	0.1555 ± 0.0005	5.0415 ±0.0720
2 kg	**37.5 %**	0.0803 ± 0.0006	3.3864 ±0.0132
	100 %	0.1878 ± 0.0013	6.1323±0.0319
3 kg	**37.5 %**	0.1200 ± 0.0044	3.7053 ±0.0205
	100 %	0.1657 ± 0.0005	5.4209 ±0.0339

Com os dados da Tabela 5.4 constata-se que a redução da rigidez no controle de posição tem o potencial de reduzir os valores de pico de corrente, sendo que em todos os casos mostrados, a redução foi inferior a metade, e em alguns casos menores que um terço da corrente de pico da situação com o K_p de fábrica. A Tabela 5.5 mostra que a aplicação da estratégia de controle adotada permite a redução da carga elétrica consumida e um controle de posição mais fiel à trajetória adotada. Todas essas características têm grande importância na manutenção dos servomotores, que podem se danificar com picos de correntes repetitivos, que geram calor excessivo durante algum tempo de caminhada do robô.

Nesse trabalho, optou-se por não utilizar o modelo demonstrado aqui para o robô completo, devido, principalmente à necessidade constante do cálculo da dinâmica inversa do robô, que necessitaria de um modelo que condiz com o comportamento dinâmico do robô real, e isso implicaria na utilização de um robô com folga mínima e que ainda conseguisse estimar a sua posição e orientação tridimensional através das informações de IMU. Tudo isso poderia tornar a tarefa complexa e requer um trabalho extenso para que o funcionamento final fosse satisfatório, como ocorreu em Schwarz e Behnke (2013).

Como será mostrado no subtópico seguinte, a redução da constante K_p foi importante para a estabilização dos movimentos, porém a redução excessiva mostrou que a técnica *feedforward* teria de ser aplicada para a redução dos efeitos passa-baixa na trajetória, que resultou em movimentos indesejados seguidos de quedas do robô. Para o robô real, a compensação dos efeitos de redução do K_p poderia ser feita com a utilização da variável de mudança de frequência do movimento do LIPM (o γ apresentado na Seção 4.1),

além de mudanças nas configurações do vetor de posturas de referência $\boldsymbol{B}$.

O algoritmo implementado em Matlab para os cálculos e obtenção dos gráficos desta seção estão apresentados no Apêndice D.

5.1 Mudança no ganho proporcional durante a caminhada

Com o objetivo de comparar o quanto a redução dos valores de K_p influenciam na estabilização da caminhada, e adicionalmente obter parâmetros de uma caminhada estável para diferentes tipos de terrenos para o estudo de identificação do terreno, foi elaborado um cenário no simulador Gazebo em que o robô tem à sua frente 6 tipos diferentes de terrenos, com características distintas de rigidez e atrito. O cenário é mostrado na Fig. 5.14, onde os terrenos estão dispostos à frente do robô em ordem decrescente de rigidez, conforme pode ser comparado aos respectivos valores da Tab. 5.7. Essa diminuição da rigidez conforme o robô avança pelos terrenos tem por objetivo aumentar o grau de dificuldade para a caminhada gradualmente, uma vez que se observou durante alguns testes, que o robô apresenta menos estabilidade quando caminhando em terrenos mais macios.

Os valores de rigidez na Tabela 5.7 podem ser comparados com os valores de referência obtidos por testes de laboratório feitos por Bosworth (2016), Tab. 5.8, e portanto, nota-se que os terrenos adotados abrangem materiais de características de impacto semelhantes à borracha sobre o colchão, até materiais muito mais rígidos que a borracha dura, ou seja, pode-se dizer, que os terrenos 5 e 6 poderiam ser comparados com pisos cerâmicos ou de concreto. Esses terrenos tiveram valores de rigidez elevados quando comparados aos valores da Tabela 5.8 , e foram escolhidos para testar a eficiência do processo de caracterização do terreno descrito no Capítulo VI, já que valores de rigidez elevada tendem a causar comportamentos dinâmicos visivelmente parecidos durante a caminhada do robô, e desejou-se verificar se mesmo com essa aparente semelhança, o método abordado consegue fazer a correta caracterização do terreno.

Os valores da constante de amortecimento K_d não foram variados nestes testes, e isso foi motivado pela constatação feita por Bosworth (2016) de que o amortecimento do terreno não é uma grandeza fácil de mensurar e que em terrenos reais ela não apresenta valores expressivos como a rigidez.

O atrito estático ($\boldsymbol{\mu_1}$) e de Coulomb ($\boldsymbol{\mu_2}$) foram selecionados de forma a poder comparar terrenos com baixo atrito, como no caso do terreno 4, com os demais terrenos de valores de atrito elevados. Além disso, o terreno 4 mostrou ser um obstáculo desafiador para o movimento do robô pois os pés durante a caminhada deslizavam constantemente.

Tabela 5-7 – Valores característicos de cada terreno utilizado para testes de caminhada e posteriormente para a identificação. Os valores se referem à constante de rigidez do terreno (K_p), sua constante de amortecimento (K_d), e os atritos estáticos e dinâmicos, μ_1 e μ_2.

	K_p	K_d	μ_1	μ_2
Terreno 1	20 KN/m	1 Ns/m	100	50
Terreno 2	50 KN/m	1 Ns/m	100	50
Terreno 3	300 KN/m	1 Ns/m	100	50
Terreno 4	300 KN/m	1 Ns/m	1	0.1
Terreno 5	3000 KN/m	1 Ns/m	100	50
Terreno 6	10000 KN/m	1 Ns/m	100	50

Tabela 5-8 – Valores de referência para tipos de terrenos de teste obtidos por testes de laboratório por Bosworth (2016).

Terreno Real	K_p
Colchão	20 KN/m
Borracha sobre o colchão	50 KN/m
Borracha dura	300 KN/m

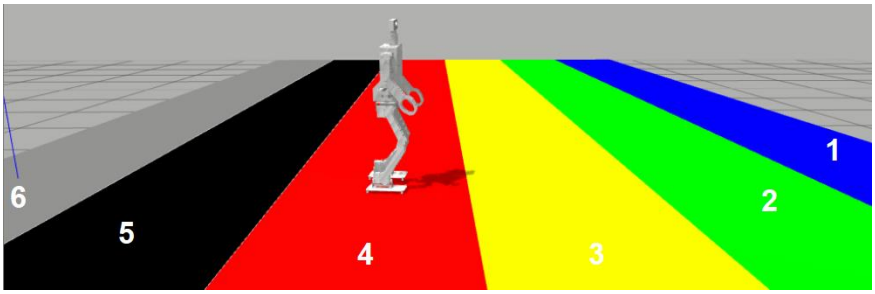


Figura 5.14 – Cenário construído para a simulação no Gazebo para seis diferentes tipos de terrenos. O tamanho de cada terreno é de 1 m de comprimento por 20 m de largura.

Os testes foram realizados para a avaliação de 4 valores de ganho proporcional K_p do controle PID de todos motores do robô simulado, sendo eles: 600, 200, 100 e 40. O maior valor de K_p = 600, foi baseado nos valores utilizados de alguns robôs simulados no simulador Gazebo. Este valor não pode ser comparado diretamente com o valor de K_p configurado para os servomotores reais, já que este último é modificado internamente por alguma constante interna desconhecida que não é fornecida pela fabricante antes de ser efetivamente aplicado ao controle do sistema (SCHWARZ; BEHNKE, 2013). Os testes foram

feitos com simulações de caminhada de dez minutos para cada K_p e foi considerado que quando o ângulo fornecido pelo IMU em torno do eixo Y, fornecesse valores absolutos maiores que 30 graus, o robô não conseguirá estabilizar a caminhada e sofrerá queda. Esse ângulo em torno do eixo Y está relacionado à inclinação para frente, ou para trás, e ele foi escolhido como parâmetro para queda pois, após muito testes tanto no robô real, quanto no simulado, a queda do robô comumente é frontal ou para trás, e raramente lateral. Os valores angulares foram derivados no tempo numericamente para a obtenção do gráfico de fase dessa variável. Todos os pontos durante os dez minutos de caminhada para cada valor de K_p do motor são mostrados na Fig. 5.15, e os dados numéricos extraídos estão apresentados na Tabela 5.9.

Tanto na Figura. 5.15 quanto na Tabela 5.9 percebe-se que a mudança da constante K_p atinge diretamente a quantidade de vezes que o robô sofre quedas e sua distância percorrida, ou seja, afeta significantemente a estabilidade durante a caminhada. Nota-se que os parâmetros com pior desempenho são aqueles com o K_p igual a 600 e o igual a 40, e para cada um concluiu-se os motivos de baixa performance. Para K_p = 600, o impacto durante a caminhada gera vibrações significativas que impedem o movimento do robô de ser executado de forma adequada, e isso provocou as constantes quedas, e o robô não conseguiu sequer completar o trajeto entre todos os seis terrenos. O valor de K_p = 40 por outro lado não conseguiu realizar o movimento de forma correta devido ao efeito passa-baixa que prejudicou a amplitude e o tempo correto de execução, e isso provocou constantes colisões entre a ponta do pé do robô com o chão, causando quedas, e da mesma forma que no caso anterior, esta configuração não conseguiu concluir a trajetória completa entre todos os terrenos. Provavelmente, com a aplicação da técnica *feedforward* apresentado neste capítulo, poderia ter melhorado o desempenho neste último caso.

Os vídeos de partes das simulações realizadas podem ser visualizados em: https://drive.google.com/open?id=164ZglWQniZqJJnqqzhErZd3vtWz2pNXb (K_p = 40), https://drive.google.com/open?id=1YVc23pZfogHgbMf-vqgqkyO9WzFN_a8h (K_p = 100), https://drive.google.com/open?id=1dWOjACLrLv46n0TDdLiElF7yDHIsTLG0 (K_p = 200), https://drive.google.com/open?id=1xl4gKMwvFeph8ssGRHsLOsu6__i6ud6C (K_p = 600).

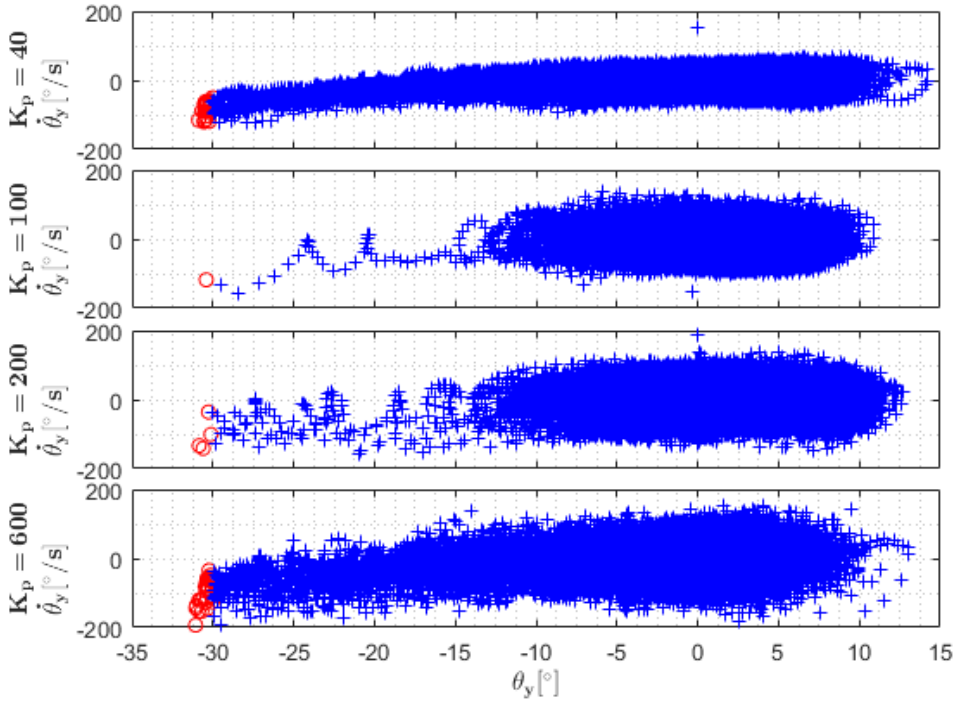


Figura 5.15 – Resultados obtidos para diferentes valores de K_p durante dez minutos de caminhada do robô nos diferentes tipos de terrenos. Os símbolos azuis em formato de cruz são pontos relacionados à caminhada estável, já os símbolos vermelhos em formato de círculo, são os pontos de instabilidade que levaram o robô a cair.

Tabela 5-9 - Resultados numéricos do experimento.

Kp	N. Quedas	N. Completos	Distância total percorrida [m]	Distância média percorrida [m]	Intervalo de Confiança para a média (95%) [m]
40	25	0	49.17	1.89	±0.58
100	1	13	80.78	5.59	±0.81
200	4	10	78.86	5.48	±0.39
600	32	0	59.03	1.78	±0.32

Para os valores de K_p de 200 e 100, houve um equilíbrio entre a absorção de impactos durante a caminhada e pouca interferência do efeito passa-baixa. Em ambos casos, o robô conseguiu atravessar todo o cenário de forma equivalente, como indicado na Tabela 5.6, e em consequência tiveram os maiores valores de distância percorrida total e média. O valor K_p de 200, no entanto, apresentou número de quedas relativamente superior para o valor de K_p de 100, o que demonstra que um valor de rigidez ótimo tende para um intervalo próximo à 100. O terreno 4, que tem os menores coeficientes de atrito, foi o que

visualmente proporcionou mais situações de desestabilidades na caminhada, como pode ser verificado nos vídeos das simulações, com exceção para o caso de K_p = 40, em que o robô tem dificuldades até mesmo de iniciar a caminhada no terreno 1.

Portanto, a mudança dos valores de rigidez do controle K_p devem ser considerados para a redução de distúrbios externos durante a caminhada do robô, sendo que valores menores que aqueles configurados para minimizar os erros de posição, podem ser benéficos quando a interação com o ambiente é um fator muito relevante como no caso dos robôs humanoides. Além disso, para reduções relevantes da constante K_p, como foi o caso do valor 40, pode ser necessária a aplicação de técnicas como a de *feedforward* para a compensação de erros de posição que degradam o movimento.

O capítulo seguinte utiliza nos testes realizados para identificação de terreno o valor de K_p = 100, que demonstrou menor quantidade de quedas.

CAPÍTULO VI

CARACTERIZAÇÃO DO TERRENO

Durante a caminhada, o ser humano apresenta a capacidade de adaptar seus movimentos de acordo com o tipo de terreno em que pisa. Essa adaptação é importante para evitar quedas e/ou esforços adicionais gerados por possíveis desequilíbrios, ou seja, essa constante caracterização e adaptação influencia diretamente na eficiência energética do movimento (DAI; TEDRAKE, 2016).

Essa mesma habilidade é desejável para a robótica móvel, uma vez que tanto eficiência energética quanto uma mobilidade robusta são importantes para viabilidade de um projeto nessa área. Por essas razões, várias pesquisas relacionadas à locomoção de robôs têm abordado a relação entre o comportamento do robô diante do ambiente de locomoção (BOSWORTH, 2016).

Como foi apresentado no Capítulo V, o robô foi simulado em 6 terrenos diferentes, e foram obtidos parâmetros tanto cinemáticos, quanto de rigidez do controle PID que pudessem proporcionar um comportamento dinâmico que consiga manter o equilíbrio durante a caminhada por estes diferentes terrenos. No entanto, não há uma garantia de estabilidade para a locomoção para ambientes diferentes dos simulados, e, portanto, conseguir classificar o terreno e adequar as estratégias de controle permitiria melhorar a robustez da caminhada e a eficiência energética.

Este Capítulo faz uma revisão a respeito dos estudos relacionados à caracterização de terreno em robótica móvel, e apresenta em detalhes a abordagem utilizada para a classificação.

Basicamente, identificar terrenos através de um robô é uma tarefa de analisar os dados provenientes dos seus sensores.

A Figura 6.1 mostra uma visão geral que todos os processos de identificação geralmente possuem (GIGUERE, 2010). No nível mais abaixo estão os sensores cujos dados são utilizados para a parte de pré-processamento, e os dados podem ser obtidos de forma contínua ou cíclica, ou seja, toda vez que o robô, por exemplo, pisa no chão, como é

comum para robôs com pernas.

O pré-processamento do sinal, tem o papel de extrair informações estatísticas relevantes, como a média, desvio padrão, curtose e/ou outras características que forem consideradas importantes. Em seguida, essas informações pré-processadas têm suas dimensões reduzidas para facilitar a tarefa de classificação da técnica utilizada para identificação do terreno. Essa redução de dimensão é feita pelo nível de extração de características, em que são aplicadas técnicas apropriadas para tal função.

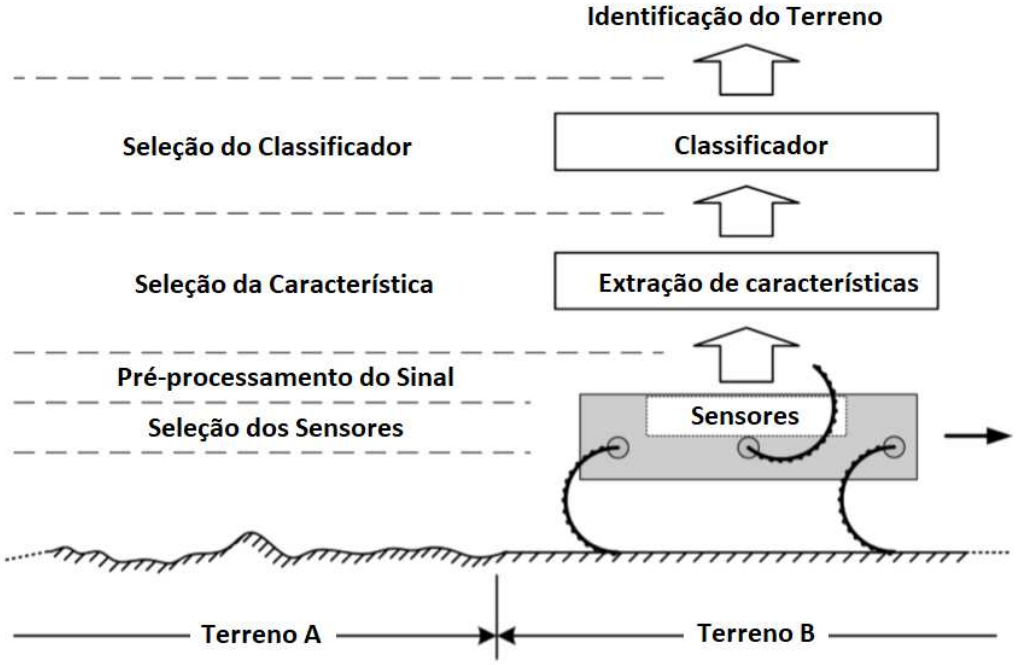


Figura 6.1 – Modelo para a identificação de terrenos (Adaptado de Giguere, 2010).

O nível de classificação aplica técnicas probabilísticas ou determinísticas para a avaliação das características extraídas, e fornecem como saída qual terreno o robô está se locomovendo. Giguere (2010) observa que a modelagem de terrenos reais é algo difícil de se obter, pois suas propriedades podem influenciar de forma significativa a resposta dos sensores, e isso torna a tarefa de identificação algo complexo de ser resolvido analiticamente. Dessa forma, Giguere (2010) conclui que a identificação de terrenos é adequada para ser resolvido de forma probabilística, pois desse modo as incertezas entre os terrenos estimados e o terreno real são levadas em consideração.

6.1 Revisão de trabalhos na área

Robôs móveis terrestres podem se locomover por meio de rodas, ou mesmo com pernas como no caso dos robôs humanoides. A maneira como o robô interage com o ambiente e adquire os dados através dos sensores, influencia diretamente na abordagem utilizada para a identificação e a acurácia obtida.

Alguns trabalhos na área de identificação de terrenos para robôs com pernas serão aqui citados conforme a configuração de seu corpo.

6.1.1 Robôs quadrúpedes

Hoffmann et al. (2014) utiliza algoritmos de *Machine Learning*, em que é utilizado *naive Bayes* e SVM (*Support Vector Machines*) para valores obtidos do movimento do torso e das articulações. Este método geralmente identifica a transição de terrenos após múltiplos passos de um robô quadrúpede e a transição entre os tipos de terrenos não é considerada.

Visão computacional é geralmente utilizada para a distinção entre as transições de terrenos e para mapear a geometria dos terrenos. Filitchkin e Katie (2012) e Fallon (2015) utilizam câmeras para a classificação de diferentes tipos de terreno em robôs quadrúpedes.

O trabalho de Bosworth (2016) detecta o tipo de solo através da resposta do controle de impedância, o que gera uma “assinatura” característica quando avaliado num curto intervalo de tempo. Este trabalho se destaca por apresentar a caracterização baseada nos princípios físicos do impacto do robô com o chão, que possibilita uma detecção de terreno e troca de controle rápida para que o robô continue sua caminhada sem sofrer quedas, incluindo as transições entre terrenos.

6.1.2 Robôs Hexápodes

Ordonez (2013) identifica o terreno por meio de medidas provenientes do sensor de inércia IMU e do feedback dos motores para ser aplicados em robôs do tipo RHex, que são robôs com 6 pernas. É utilizada a frequência do movimento do corpo para classificar a locomoção para diferentes ambientes. Assim que o robô "sente" a mudança ele modifica a impedância da perna. Best (2013) utiliza *feedback* da cinemática da perna durante o movimento de um robô de seis pernas, e métodos estatísticos são aplicados para a determinação do terreno, porém não são analisados os princípios físicos.

6.1.3 Robôs Humanoides

Nishiwaki et al., (2012) mostra um sistema de navegação para um robô humanoide baseado na percepção do solo por meio de sensores lasers.

Walas; Kanoulas e Kryczka (2016) faz uso de sensores de torque e força nos pés do robô para a classificação e mudança de estratégias de locomoção em um robô humanoide. Para isso ele realiza a transformada discreta de wavelet ou DWT (do inglês *Discrete Wavelet Transform*) nos dados provenientes dos sensores e em seguida utiliza SVM (*Support Vector Machines*) para a classificação dos dados obtidos. Neste trabalho é alcançada uma acurácia de 95 % com a abordagem adotada.

Yuan e Wang (2017) desenvolveram e testaram um sensor táctil resistivo que foi preso à sola do pé do robô humanoide NAO e fornece dados durante a caminhada do robô. Os dados gerados são processados pelo algoritmo *k-Nearest Neighbor* (kNN) que consegue diferenciar até quatro terrenos com uma precisão de 95 % em até 4 passos após a transição entre terrenos.

6.2 Motivação para o método utilizado

Um dos métodos citados que motivou a abordagem utilizada nessa dissertação, é o de Bosworth (2016), em que o robô quadrupede *Super Mini Cheetah* ou SMC, consegue distinguir o tipo de terreno que está pisando em um tempo relativamente curto, segundo o autor em torno de 45 ms. A técnica utilizada é a análise da assinatura do terreno através da resposta ao controle de impedância da perna do robô. A Figura 6.2 mostra a resposta do controle no momento do impacto durante a caminhada do robô em diversos tipos de terreno, e essas curvas caracterizam a forma como o impacto em cada terreno interfere no controle de torque dos motores do robô, e devido à distinção entre essas curvas, elas foram chamadas de assinatura do terreno. Bosworth (2016) baseia o seu modelo de impacto entre o robô e o terreno no modelo de “um quarto de carro” que é utilizado extensivamente no desenvolvimento da suspensão de automóveis (HROVATT, 1997). O modelo de impacto obtido serve como apoio teórico para a técnica de classificação de terrenos desenvolvida, em que é feita uma integral no tempo dos valores do controle de impedância durante um período do impacto, e com essa informação ele mostra que foi possível fazer uma clara distinção entre 5 tipos de terrenos.

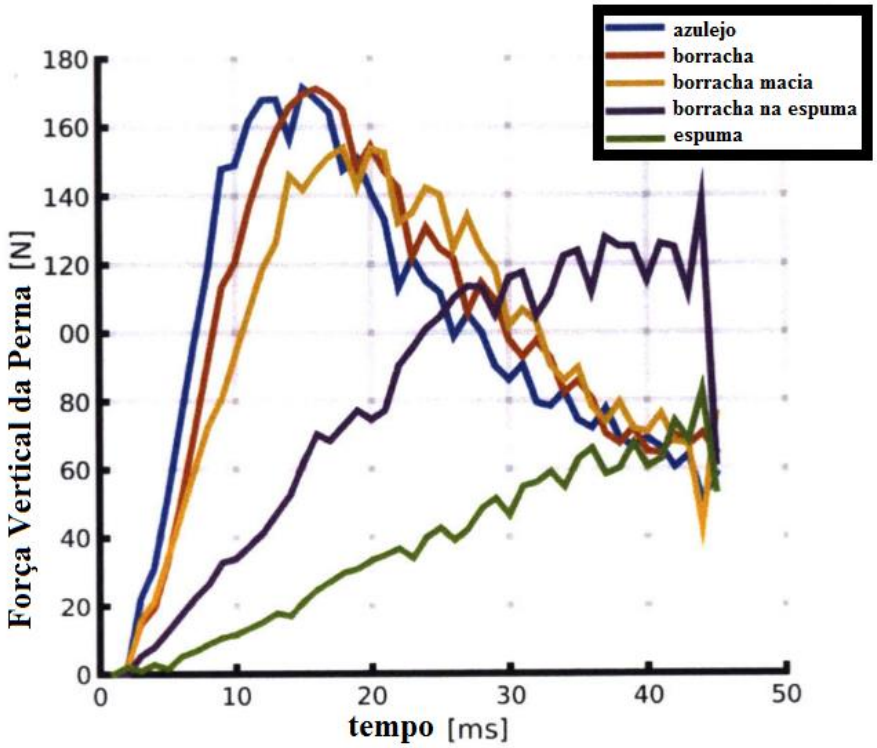


Figura 6.2 – Resposta do controle de impedância para diferentes tipos de terrenos realizado pelo robô Super Mini Cheetah (adaptado de BOSWORTH, 2016).

A estratégia de controle da locomoção utilizada por Bosworth (2016) é de obter empiricamente parâmetros de controle que estabilizam a caminhada do robô SMC para diferentes tipos de terreno. Dessa forma, o robô consegue modificar sua caminhada de acordo com o tipo de piso detectado pela técnica citada. Os resultados obtidos mostram que o robô consegue manter um caminhar estável mesmo com mudanças repentinas de terreno.

Como foi mostrado no Capítulo V através de testes em simulação, o robô humanoide estudado demonstra comportamento promissor na sua estabilização com a mudança do ganho proporcional dos motores da perna. Além disso, a estratégia utilizada para a geração de trajetórias, o LIPM, com as modificações posturais e na frequência de execução, mostrou resultados positivos para uma caminhada com geração online de movimento tanto para o robô real quanto para o robô simulado.

A leitura do torque exercido pelos servomotores pode ser inferida da leitura das correntes dos motores que é obtido diretamente dos seus registradores durante sua movimentação, e no caso da simulação o torque pode ser obtido diretamente através das informações disponíveis pelo simulador Gazebo.

Contudo, algumas observações devem ser feitas ao considerar a aplicação do método de Bosworth (2016) em um robô humanoide:

- É possível fazer a leitura de torque de ambos robôs, o que facilita na análise da resposta de impedância quando o robô atinge o solo.
- Ambos são robôs com pernas e o impacto com o terreno influencia diretamente na performance na locomoção.
- O fato de o robô humanoide ter duas pernas ao invés de quatro, como no caso do SMC, faz com que as características dinâmicas do movimento sejam diferentes, e o robô tenha uma tendência a cair com uma facilidade maior. Momentos de desequilíbrio no robô humanoide tem o potencial de causar respostas de impedância muito distintas para um mesmo terreno, o que agrava a identificação do terreno com a análise da curva de resposta do torque.
- O robô humanoide tem como efetuador um pé de dimensões consideráveis em relação ao resto da perna, e no caso do robô SMC o seu efetuador pode ser considerado praticamente pontual. Este fator modifica a resposta do controle, uma vez que o pé no caso do robô humanoide tende a contribuir para a estabilização do movimento ao custo de torques superiores nos motores do tornozelo e do calcanhar, e novamente, isso modifica a resposta ao torque em momentos de desestabilização. Já no robô SMC, a resposta durante a caminhada segue um padrão bem definido para um mesmo terreno, e isso contribui significativamente para a determinação do terreno através das curvas do controle de impedância.
- O robô humanoide proposto e utilizado nesta dissertação não conta com um sensor de força no pé para determinar o momento em que o pé atinge o solo como no caso do SMC. Esse momento de impacto é importante para a determinação do início e fim da curva de resposta. Apesar disso, é possível estimar esse momento com base nos comandos enviados ao pé por meio das informações do LIPM durante a execução.
- O robô humanoide apresentou problemas de assimetria durante o movimento, mesmo no modelo simulado, e isso foi constatado também do robô real. Essa assimetria tem o potencial de causar respostas de impedância diferentes para cada pé que toca o chão, e isso pode dificultar a caracterização do terreno.

Apesar dessas diferenças serem limitantes para a aplicação do método de Bosworth (2016), ainda assim a ideia da busca de uma assinatura do terreno foi mantida, e para isso levou-se em consideração a utilização de informações adicionais para complementar as informações de impedância da perna. Para isso, utilizou-se os dados de giroscópio, acelerômetro e ângulos obtidos do IMU, que podem ter seus dados relacionados aos momentos de desestabilização da caminhada. Com essas novas informações é esperado que se consiga correlacionar os efeitos do desequilíbrio com o torque que é provocado em

determinadas articulações. E dessa forma, espera-se ser possível a distinção entre diferentes terrenos mesmo com a mudança de comportamento dinâmico.

Foi realizado um estudo para a distinção entre terrenos tanto para o robô simulado quanto para o robô real, sendo que para o primeiro caso, foram considerados os seis terrenos descritos no Capítulo V. Já para o robô real, foram considerados seis terrenos de características de rigidez distintas: edredom, grama sintética, borracha, carpete, MDF e cerâmica.

Para alcançar o objetivo de classificação, foi necessário, primeiramente, determinar qual o período de amostragem necessário quando o pé do robô atinge o solo. Os sinais escolhidos para realizar a amostragem foram o torque no joelho e no calcanhar, e as informações de ângulos do giroscópio, acelerômetro e ângulos medidos (Figura 6.3). A escolha do local de leitura de torque foi feita com base em testes sucessivos de caminhada, em que se constatou visualmente que estes locais sofrem uma influência maior do tipo de terreno simulado durante a caminhada, e possuem uma uniformidade maior de suas curvas durante a locomoção.

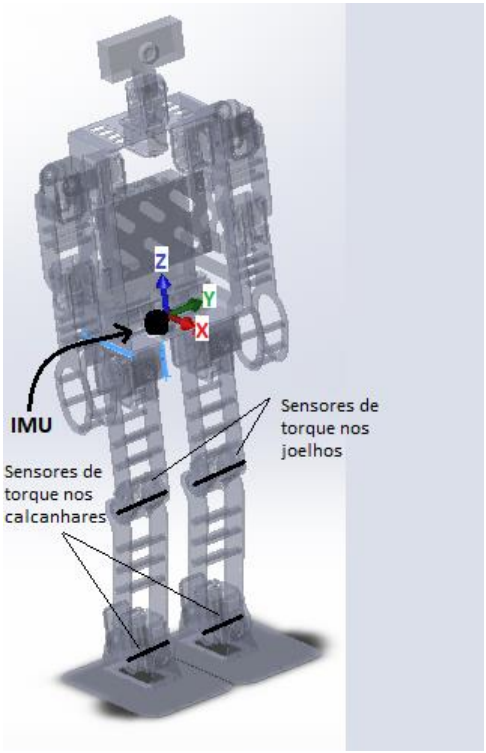


Figura 6.3 – Localidades dos sensores utilizados para a identificação do terreno e o sistema de referência relativa ao robô que será utilizada ao longo desse capítulo.

Os parâmetros de caminhada são os mesmos utilizados no Capítulo IV, e foi possível

perceber durante os testes de caminhada que o tempo de 0.244 s é adequado para captar o efeito do impacto durante um passo tanto para o robô real quanto para o robô simulado. Esse tempo determinou a quantidade de pontos necessários a ser coletada para cada sensor, sendo que tanto os sensores de torque quanto o IMU foram configurados para uma taxa de amostragem de 125 Hz, o que fornece a quantidade de 28 pontos durante o período de impacto. Essa frequência de amostragem foi baseada nos valores máximos de taxa de atualização do sensor IMU utilizado no robô humanoide, o Phidgets (PHIDGETS, 2016) , e o mesmo valor de frequência foi adotado para a atualização do torque por padronizar a forma de aquisição dos dados no tempo durante o momento de impacto, e por ser possível obter essa taxa de dados na leitura dos servomotores que tem taxa de transmissão de dados de 1Mbps.

Com base nesse período de aquisição e nas informações selecionadas para serem lidas, foi elaborado um algoritmo que coleta todos esses pontos para cada vez que o módulo do LIPM indica que um pé está no chão. Além disso, o LIPM indica qual pé está no chão, o que possibilita coletar apenas as informações do pé de impacto a cada momento. Foram coletados 5 minutos de dados para cada terreno, tanto para o robô real, quanto para o robô simulado, o que correspondeu em média à 800 passos no total, sendo 400 para cada perna. O vídeo de parte da coleta de dados para o robô real pode ser visualizado em: https://drive.google.com/open?id=1jE-5qiClNDwqDHVmn0gsfYblApSczxRY.

As Figuras. 6.4 e 6.5 mostram os perfis de impacto para cada terreno, levando em consideração 28 pontos de amostragem para os diferentes terrenos para o robô real e simulado em ordem de rigidez da esquerda para direita. Para o robô real foi escolhido um valor de velocidade de caminhada $v_x = 0$ m/s, ou seja, o robô deveria teoricamente manter-se no mesmo lugar durante os testes, e essa escolha foi baseada no fato do robô ter momentos de desequilíbrios constantes para os diferentes tipos de terrenos quando valores positivos de v_x eram selecionados. O comportamento do robô foi de manter, na maior parte do tempo, um balanço lateral do corpo, apoiando ora em um pé, ora no outro, porém, girando constantemente em torno do próprio eixo, e isso ocorreu devido às assimetrias mecânicas na perna do robô. Já para o robô simulado, a velocidade v_x foi variada de minuto em minuto durante os cinco minutos do teste, passando pelos seguintes valores nesta ordem: 0.06, 0.12, 0.18, 0.24 e 0.30 m/s. Essa estratégia foi escolhida para verificar se o método proposto para a classificação de terrenos consegue uma boa performance mesmo se o robô tiver variações na sua velocidade. As linhas sólidas representam as médias das 400 medidas obtidas para cada situação, e a região sombreada representa o desvio padrão das medidas. Os valores de ângulos em torno do eixo Z não foram levados em consideração, pois o filtro utilizado para a obtenção dos ângulos com base no IMU

apresentou divergências em relação a esse eixo quando comparado com o comportamento do robô.

Ao comparar as curvas de um mesmo sensor e terreno para a Figura. 6.4, percebe-se a falta de simetria entre ambos lados da resposta ao impacto, que pode ser causada por folgas mecânicas e ou de desempenho dos servomotores. Esse desbalanço é mais evidente entre os valores de torque, o que indica que o efeito global das assimetrias durante o movimento promove esforços significativamente diferentes em determinada perna para compensar o desbalanço de forças no sistema. Esse comportamento tem o efeito negativo de modificar a trajetória desejada para a caminhada, já que o robô tende a realizar curvas imprevistas, e, além disso, pode sofrer picos de torques maiores que os permitidos para os motores.

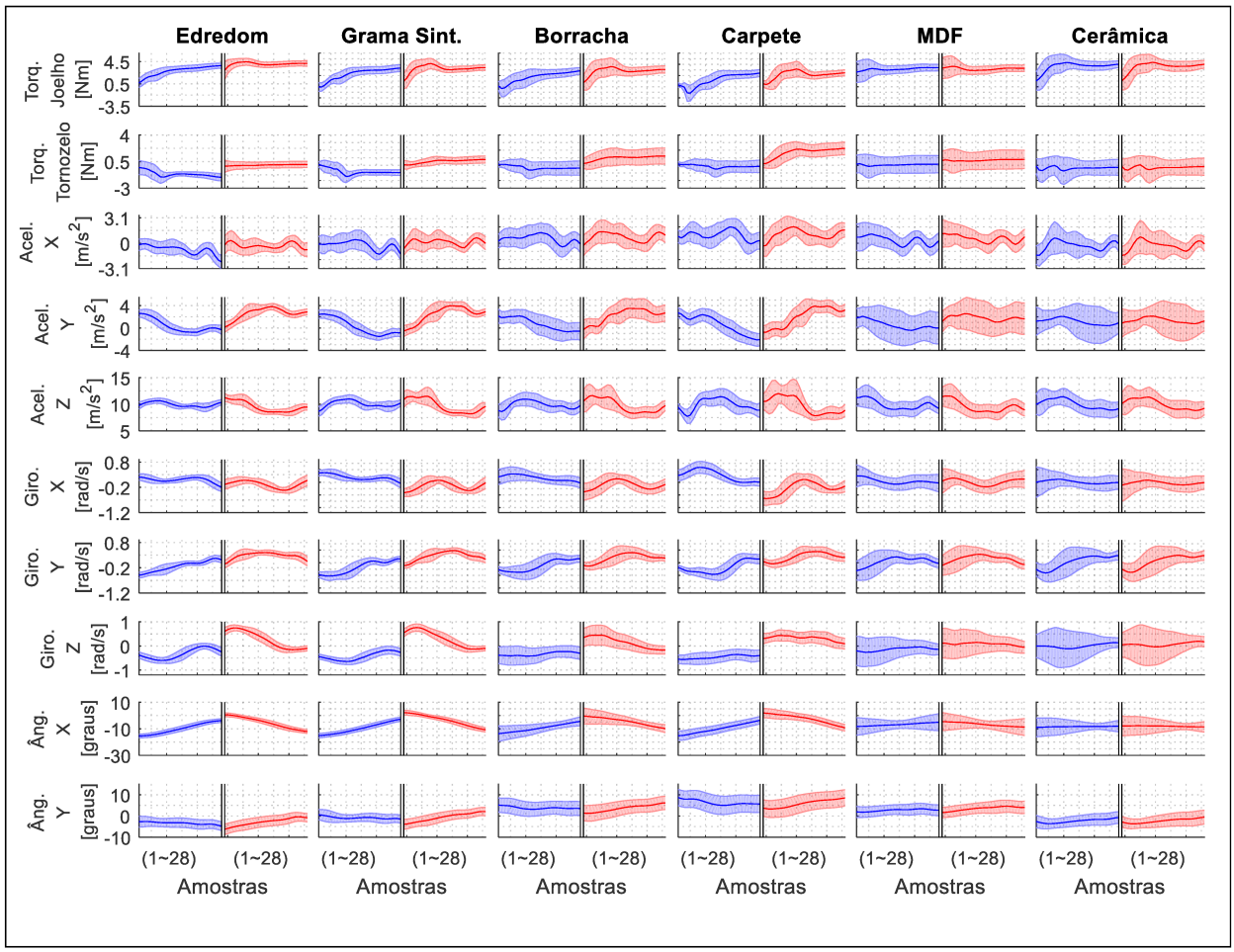


Figura 6.4 – Perfis de impacto para cada terreno para os testes reais. As linhas sólidas representam as médias e a região sombreada representa o desvio padrão das medidas. A cor vermelha corresponde aos dados do pé direito e a azul para o pé esquerdo.

Outro aspecto que pode ser observado, é que terrenos com maiores rigidez

(considerando que os terrenos estão em ordem crescente de rigidez da esquerda para a direita) tendem a gerar um desvio padrão maior nos dados dos sensores, principalmente nas informações de aceleração no eixo Y, que corresponde à oscilação lateral do robô. Esse efeito pode ser resultado do comportamento oscilatório que o robô desempenha ao tocar o chão, sendo que para terrenos mais rígidos, essa oscilação tende a ser de maior frequência e, portanto, com maior chance de gerar perfis de resposta diferente para o terreno a cada passo. Já para terrenos menos rígidos, o robô tende a ter esse efeito amortecido pelo terreno, e, portanto, as curvas de resposta apresentam menor frequência, e tem um comportamento mais bem definido.

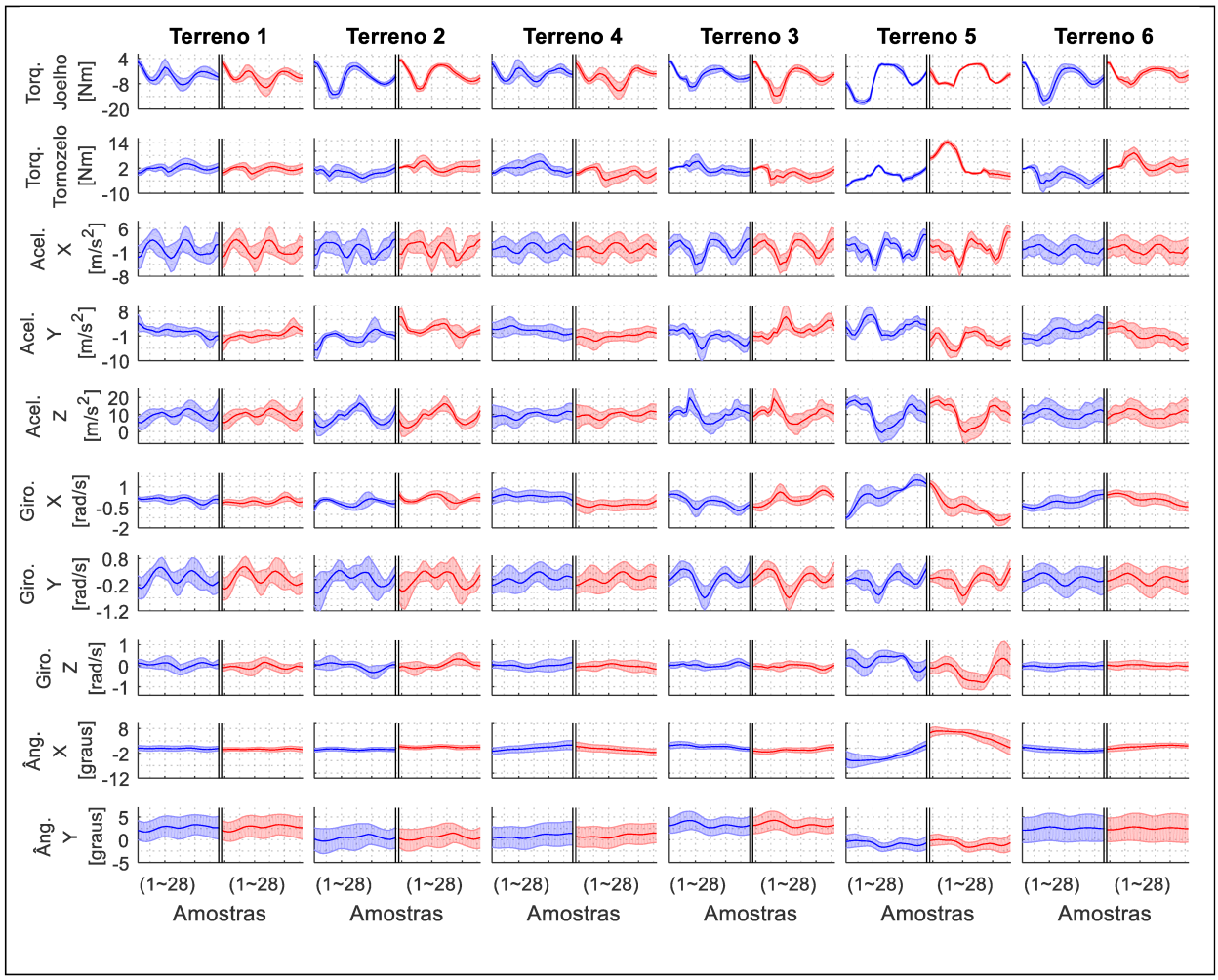


Figura 6.5 - Perfis de impacto para cada terreno para os testes simulados. As linhas sólidas representam as médias e a região sombreada representa o desvio padrão das medidas. A cor vermelha corresponde aos dados do pé direito e a azul para o pé esquerdo.

No caso da Figura 6.5, em que são mostradas as respostas para os terrenos simulados, nota-se que os efeitos de assimetria também estão presentes, e isso indica que

não apenas características mecânicas do robô real podem afetar a precisão da execução dos movimentos, mas também prováveis efeitos dinâmicos não previstos pelo LIPM que podem surgir do movimento oscilatório do robô durante a caminhada. Um desses efeitos ficou evidenciado durante testes simulados, ao perceber que durante um determinado passo com um dos pés, o tronco do robô se encontrava inclinado para frente, e no outro passo com o outro pé, o tronco se encontrava para trás. Essa mudança constante de inclinação do tronco tem o potencial de gerar torques desproporcionais para cada perna, e, portanto, deixar o movimento menos simétrico.

A procura por um padrão que relaciona os parâmetros analisados com os gráficos obtidos pelas Figs. 6.4 e 6.5 para cada tipo de sensor é uma tarefa difícil quando é levado em conta mais de dois tipos de terrenos e os momentos de instabilidade. No caso de Bosworth (2016), a integração no tempo da resposta do controle de impedância forneceu valores distintos para os diferentes tipos de terrenos, e esse método teve fundamentação física, que foi possível devido às características relativamente simples do robô SMC em comparação ao robô humanoide.

No caso do robô humanoide estudado, a quantidade de graus de liberdade é superior, e o comportamento dinâmico se torna, como consequência mais complexo, e isso dificulta realizar um embasamento teórico para a resposta ao impacto. Trata-se, portanto, de um problema que apresenta um grau de não linearidade.

É importante observar que os terrenos utilizados para a tarefa de classificação para o robô simulado, são caracterizados por parâmetros reduzidos, como uma constante de rigidez, amortecimento, atrito estático e de Coulomb. Porém, intuitivamente, sabe-se que em situações reais existem outros fatores que podem ser determinantes na resposta ao impacto, como por exemplo, no caso da grama sintética, a própria altura da grama pode influenciar em como o pé do robô lida com os possíveis desníveis gerados durante a pisada.

Portanto, com o objetivo de caracterizar os terrenos de maneira mais abrangente possível, foi levado em consideração todas as fontes de dados apresentadas, e a forma como eles foram tratados para o modelo de classificação será mostrado no tópico seguinte.

6.3 Geração da assinatura do terreno

Serão utilizadas as 10 informações de sensores apresentadas, sendo que cada uma fornece 28 pontos para cada momento que um pé do robô toca o chão, ou seja, são 280 pontos de informação no total. Para tornar a visualização desses dados mais clara, e para tentar tornar mais fácil uma caracterização visual de cada tipo de terreno, foi feito uma

imagem em que cada linha representa os dados de uma das cinco informações/leituras na seguinte ordem da primeira linha superior para baixo: torque no joelho, torque no calcanhar, aceleração linear nos três eixos na ordem XYZ, velocidade angular nos três eixos na ordem XYZ e posição angular nos eixo X e Y. As colunas são as informações no tempo, sendo que cada pixel foi colorido na escala de cinza para uma escala absoluta, onde a cor preta é o menor valor dos dados, que foi arredondado para -10, e a cor branca é o maior valor dos dados, que foi arredondado para 30.

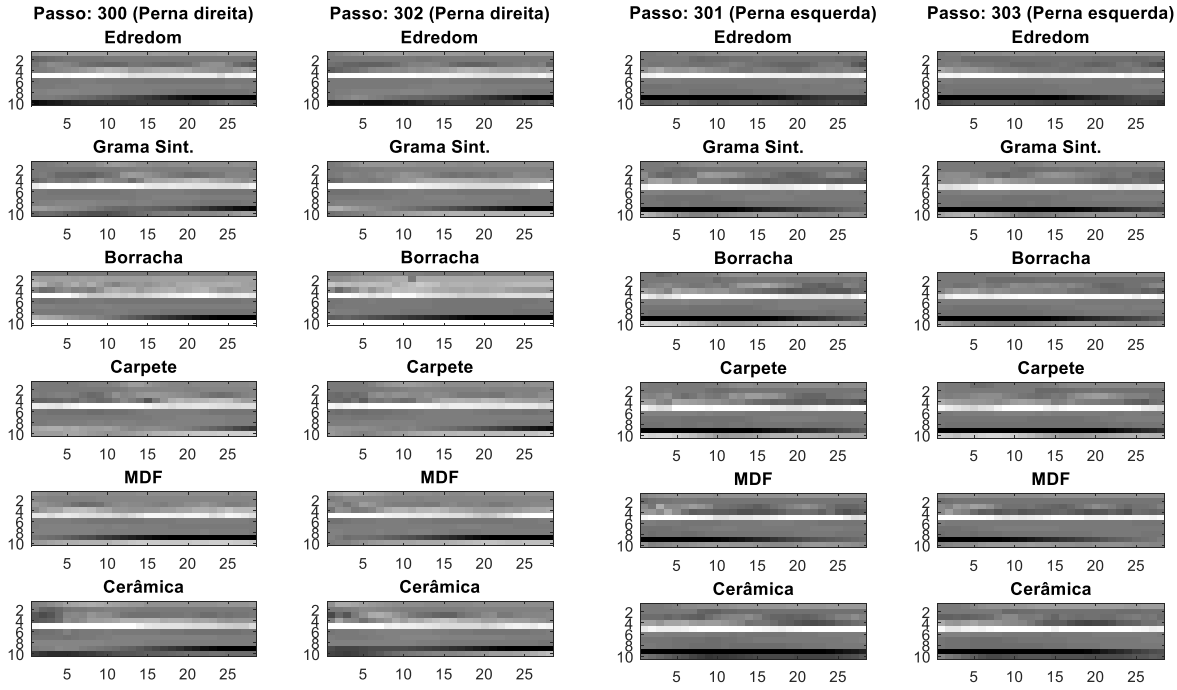


Figura 6.6 – Imagens que representam as assinaturas dos seis tipos de terrenos obtidas com dois passos sucessivos com ambas pernas.

A Figura 6.6 mostra as assinaturas de impacto geradas por dois passos consecutivos para os seis diferentes terrenos reais estudados. Percebe-se uma nítida diferença entre os dados gerados, e há uma clara semelhança entre os passos dados com o mesmo pé no mesmo terreno. As mesmas observações são feitas para ambos pés, porém, nota-se que há diferenças entre as assinaturas geradas entre eles devido às assimetrias do movimento. Portanto, com as imagens geradas introduz-se a ideia de assinatura do terreno, que foi motivada inicialmente pela abordagem de Bosworth (2016). Porém, inferir uma técnica de análise desses dados é algo complexo, já que cada assinatura produz uma imagem característica que representam curvas características dos sensores utilizados. Para realizar o processamento dessas informações, e classificação dessas imagens em relação aos terrenos correspondentes, foi utilizada as redes neurais convolucionais, que tem a

propriedade de lidar com informações em formato bidimensional de forma computacionalmente eficiente, e com propriedades que favorecem o aprendizado de dados em formato de imagem.

Nesta dissertação não foi feita a análise da detecção nos momentos de transição entre terrenos, porém, espera-se que o método tenha capacidade de realizar essa tarefa, devido à sua característica de constante análise dos dados colhidos para cada passo.

6.4 Redes Neurais Convolucionais

Redes neurais artificiais tem sido extensivamente exploradas em diversos trabalhos desde as primeiras ideias desenvolvidas nessa área na década de 1950, em que o cientista Frank Rosenblatt desenvolveu o conceito de *perceptrons*, que se trata de uma forma muito simples de representação de um neurônio humano para tomadas de decisões com base em um conjunto de entradas (NIELSEN, 2015). A Figura 6.8 mostra a representação gráfica de um *perceptron*.

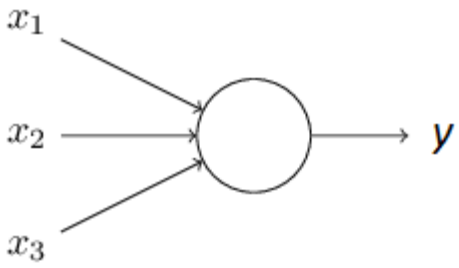


Figura 6.8 – Modelo básico de um *perceptron* com três entradas x_1, x_2 e x_3 e uma saída y (Adaptado de NIELSEN (2015)).

O *perceptron* da Fig. 6.8 possui três entradas x_1, x_2 e x_3, e Rosenblatt propôs que a saída y seria um valor binário, que se torna 1 se a soma dos valores da entrada multiplicados cada um por pesos w_i são maiores a um valor limite l, caso contrário, esse valor se torna 0. Matematicamente, o valor de saída de um *perceptron* em relação às suas entradas podem ser representados pela seguinte formulação:

$$y = \begin{cases} 0 \; se \; \sum_i w_i x_i \; \leq l \\ 1 \; se \; \sum_i w_i x_i > l \end{cases} ; \; com \; i \; de \; 0 \; \text{à} \; quantidade \; de \; entradas \tag{6.1}$$

Dependendo do que as variáveis de entrada x_i representam, e da forma como os

pesos estão configurados no *perceptron*, pode-se determinar um modelo simples de decisão. Por exemplo, as entradas na Fig. 6.5 poderiam ser respectivamente as variáveis relacionadas à média, desvio padrão e valor de pico dos dados coletados da inclinação do robô no momento de impacto com o chão, e os pesos poderiam ser escolhidos empiricamente de forma a conseguir identificar se o piso em que o robô está no momento é grama sintética ($y = 1$) ou não ($y = 0$).

Outras formas de problema também podem ser formuladas, e pode ser desejável mais de uma saída para a tomada de decisão. Para decisões mais complexas as redes podem apresentar configurações mais complexas, como a mostrada na Fig. 6.9.

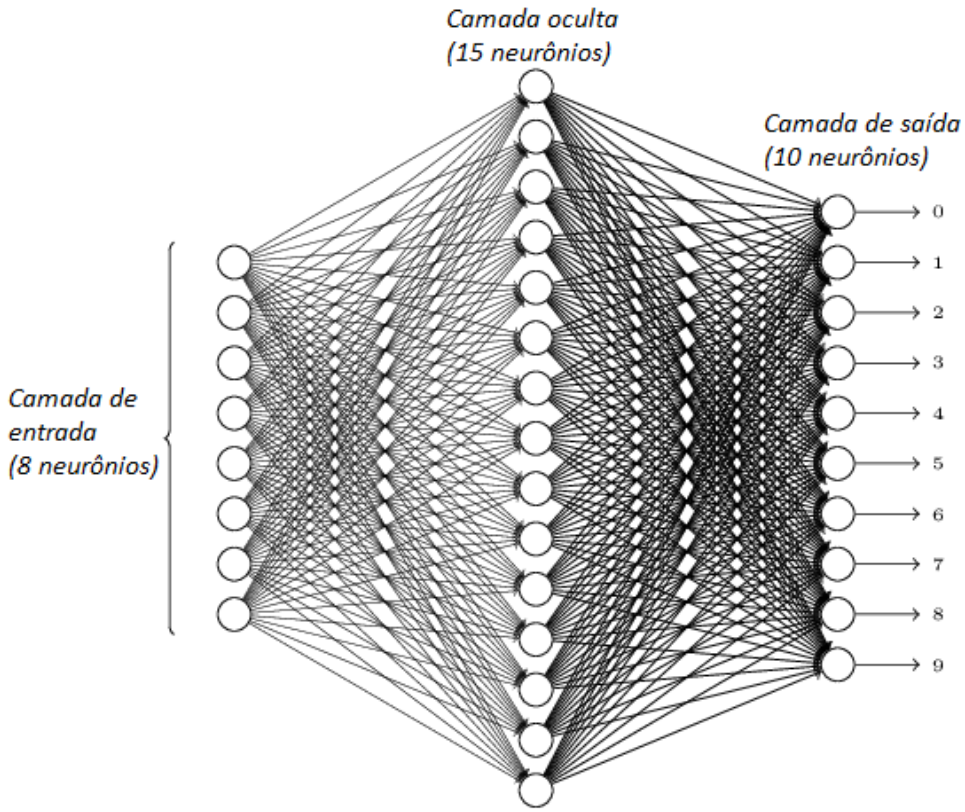


Figura 6.9 – Exemplo de estrutura para rede neural para 8 entradas, 10 saídas, e uma camada oculta (*hidden layer*) com 15 neurônios (Adaptado de NIELSEN (2015)).

A Figura 6.9 apresenta o conceito de camadas, sendo a primeira à esquerda a camada de entrada, a segunda é conhecida como camada oculta (*hidden layer*) e a terceira é a camada de saída. Nota-se que nesse tipo de configuração, a quantidade de pesos é muito grande, em que para cada um dos 8 neurônios da primeira camada existem 15 pesos, e cada neurônio da segunda camada estão ligados 10 pesos, ou seja, essa rede apresenta

8x15x10 = 1200 pesos no total. A tarefa de escolher o valor de cada peso para que a rede tome as decisões desejáveis manualmente é uma tarefa impraticável, e por isso foi desenvolvido o conceito de aprendizagem, na rede neural, em que os pesos, e qualquer outros parâmetros adicionados à rede neural podem ser determinados através de algoritmos de aprendizagem que usam como base um conjunto de dados de entrada e de saída desejado para cada entrada. Dessa forma, o algoritmo é capaz de encontrar o melhor conjunto de variáveis internas à rede neural para que esta consiga reproduzir o comportamento desejado, mesmo para entradas que não estavam previstas.

Assim como as técnicas de aprendizagem, várias abordagens foram desenvolvidas ao longo das décadas para o aprimoramento das redes neurais, e novas topologias e métodos de ativação foram desenvolvidos com o objetivo de otimizar as redes neurais para a aplicação em tipos específicos de problemas. Nesse contexto de aprimoramento da rede neural, surgiram as redes neurais convolucionais, também conhecidas como CNN (do inglês *Convolutional Neural Network*), que são um tipo especializado de redes neurais para processamento de dados com topologia do tipo grade (comumente citada como *grid-like topology*), como por exemplo dados coletados no tempo em intervalos regulares, ou mesmo imagens que podem ser consideradas como dados em uma grade bidimensional (GOODFELLOW; BENGIO; COURVILLE, 2016).

As CNNs são, basicamente, redes neurais que usam a operação de convolução no lugar de multiplicação de matrizes em pelo menos uma de suas camadas. A operação de convolução entre uma entrada unidimensional x e um conjunto de pesos w de uma camada é dada pela Eq. 6.2.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{6.2}$$

O conjunto de valores w são denominados de núcleo (*kernel*), e a saída da convolação é geralmente chamada de mapa de características (*feature map*). O valor de $s(t)$ é o resultado da convolução entre a entrada x e os pesos w, e a variável a é uma variável auxiliar para percorrer todos os valores de entradas dentro do somatório e realizar a convolução. Esses mesmos conceitos podem ser aplicados para conjunto de dados bidimensionais, como por exemplo uma imagem, em que cada pixel é um valor dado pela função $I(i,j)$ e como no caso anterior, a operação de convolução é feita com um núcleo, mas agora bidimensional $N(i,j)$, e a operação fica:

$$S(i,j) = (N * I)(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} I(i-m, j-n) N(m,n) \tag{6.3}$$

Da mesma forma que na Eq. (6.2), $S(i,j)$ é o resultado da convolução ao longo da imagem $I(i,j)$ com o núcleo $N(i,j)$. Os valores m e n, fazem o papel da variável a na Eq. (6.2), mas nesse caso essas variáveis auxiliam a percorrer toda a imagem $I(i,j)$, e por isso o duplo somatório que tem como limites as dimensões máximas M e N da imagem.

A introdução da operação de convolução nas redes neurais melhora o sistema de aprendizado devido à três propriedades importantes (GOODFELLOW; BENGIO; COURVILLE, 2016):

- Interação Esparsa: redes neurais tradicionais trabalham com a multiplicação da matriz de parâmetros da rede neural (como os pesos), sendo que há um parâmetro para cada entrada e saída do sistema, ou seja, cada saída interage com cada entrada do sistema. Já no caso das redes convolucionais, há uma interação esparsa entre entrada e saída, uma vez que o núcleo é menor que a entrada, e isso faz com que a quantidade de parâmetros necessários para coletar informações relevantes do sistema seja bem menor do que no caso das redes tradicionais. Além disso, essa interação esparsa requer menos capacidade computacional, já que necessita de menos cálculos efetuados. A Figura 6.10 demonstra essa ideia graficamente.
- Compartilhamento de parâmetros: essa propriedade refere-se ao uso do mesmo parâmetro na rede para mais de uma entrada, ao contrário das redes tradicionais em que cada peso por exemplo, está associado à apenas uma entrada e é utilizado apenas uma vez.
- Equivariância: o compartilhamento de parâmetros em conjunto com a operação de convolução promove a equivariância da translação. Isso significa que se a entrada sofre uma mudança, a saída também mudará da mesma forma.

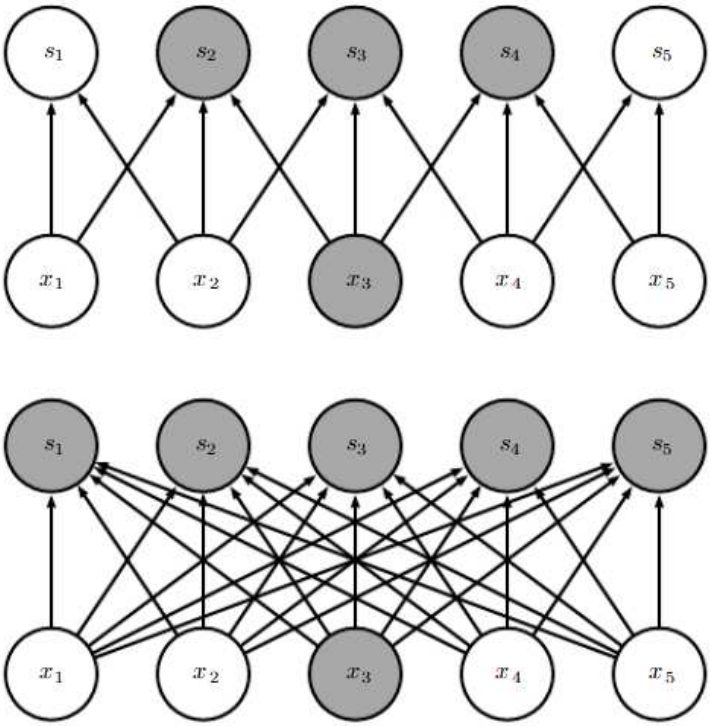


Figura 6.10 – Conectividade esparsa. Os elementos x_i são as entradas, e os elementos s_i as saídas. É mostrado quais elementos são influenciados pela entrada x_3 tanto para o caso da rede convolucional na parte superior, quanto para a rede tradicional na parte inferior. Os elementos em cinza são aqueles que apresentam algum meio de conectividade. No caso da rede convolucional com núcleo de tamanho 3, apenas os elementos s_2, s_3 e s_4 são afetados pela entrada x_3, e o número de conexões é bem menor entre as duas camadas. Já no caso da rede tradicional, a entrada x_3 afeta todas as saídas, e apresenta como consequência muito mais conexões a consequentemente cálculos entre as duas camadas.

Uma camada de rede neural convolucional pode consistir de três estágios:

- As operações convolução conforme foi apresentada, em que um núcleo executa as operações da Eq. (6.3) para as entradas.
- Em seguida os valores obtidos nas convoluções, se tornam a entrada de uma função de ativação não-linear, como por exemplo a função de ativação retificada linear ReLU (do inglês, *rectified linear activation function*), em que para um determinado valor x o valor dessa função $f(x)$ é dada por : $f(x) = max(0, x)$. Esse estágio é conhecido por estágio de detecção.
- O terceiro estágio é conhecido por *pooling*, em que para as saídas do estágio anterior, são selecionados diversos grupos de valores, em que é selecionado como saída apenas o maior deles. Essa operação tem o potencial de tornar a camada convolucional invariante a pequenas translações da entrada (GOODFELLOW;

BENGIO; COURVILLE, 2016), ou seja, no caso dos dados das assinaturas obtidos (Figuras 6.6 e 6.7), mesmo se houver atraso na coleta dos dados, isso pode não ter grande influência na caracterização, pois essa translação no tempo dos dados não gera diferenças significantes nas camadas convolucionais da rede.

Portanto, devido às características apresentadas das CNNs, como a capacidade de lidar com processamento de imagens sem o aumento significativo de parâmetros, e as propriedades de invariância à translação fazem a escolha das redes neurais convolucionais adequadas para o processo de classificação das imagens com os dados de impacto obtidos ao longo do tempo pelo robô humanoide.

6.5 CNN utilizada

A estrutura da CNN utilizada para a classificação tanto dos terrenos reais quanto dos simulados teve a seguinte estrutura de camadas na ordem descrita:

- Camada de entrada para a matriz que corresponde à assinatura do terreno durante o impacto (Figuras 6.6) de dimensões 10 linhas por 28 colunas.
- Camada convolucional de dimensões 3 linhas por 3 colunas e 16 filtros.
- Camada de pooling 2 linhas por 2 colunas para cada um dos 16 filtros.
- Camada convolucional de dimensões 3 linhas por 3 colunas e 32 filtros.
- Camada de pooling 2 linhas por 2 colunas para cada um dos 32 filtros.
- Camada de 12 neurônios totalmente conectados com a camada anterior.
- Camada com 6 neurônios, cujas saídas foram normalizadas para valores entre 0 e 1, para representar a probabilidade de determinado terreno ser o selecionado entre os 6 propostos tanto nos testes reais quanto nos simulados. Por exemplo, uma saída do tipo: [0.1 0.1 0.1 0.7 0.1 0.0], significaria que com 70 % de certeza que as entradas fornecidas na primeira camada representam o impacto gerado pelo carpete, ou pelo terreno, no caso da simulação.

A Figura 6.11 mostra um esquema da rede neural utilizada, em que a imagem do impacto é a entrada da rede, e a saída determina a possibilidade de detecção do terreno.

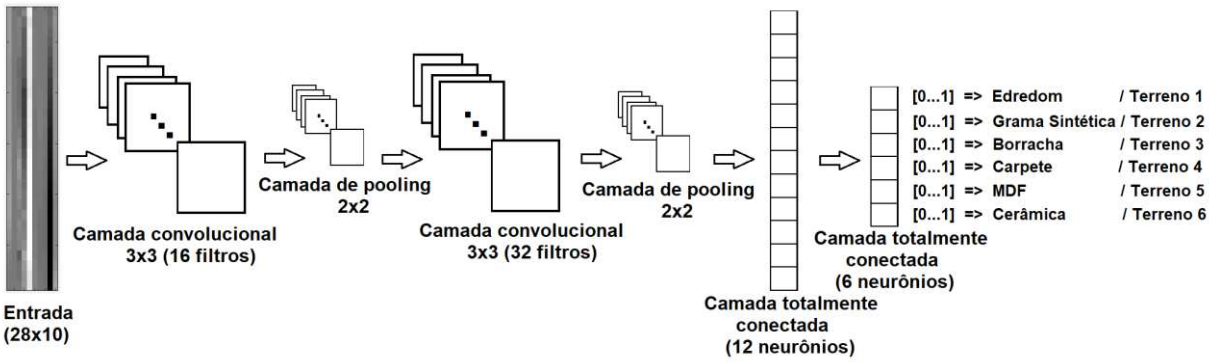


Figura 6.11 – Ilustração das camadas da rede neural convolucional utilizada.

Para o processo de aprendizagem, foi utilizado a otimização Adams, que tem o papel de atualização dos parâmetros da rede neural, como os pesos, para atingir valores que atendam ao propósito de classificação. A descrição deste método de atualização dos pesos é explicada em Kingma e Ba, (2014). Ele foi configurado com o valor de taxa de aprendizagem de 0.001. A taxa de aprendizagem utilizada para o otimizador Adams foi o valor padrão utilizado para modelos no geral, já configurado na biblioteca TFLearn utilizada.

A estrutura da rede neural foi escolhida com base na configuração utilizada para identificação de texto em documentos no trabalho de Lecun *et al.*, (1998). A rede teve alguns dos parâmetros reduzidos para que pudesse ser encontrada uma solução mais compacta e eficiente para a classificação dos terrenos.

Foi aplicada também a técnica conhecida por *dropout* para a camada totalmente conectada, em que com uma certa probabilidade, alguns neurônios são desativados por alguns momentos no momento de aprendizagem, e, portanto, seus valores passam a não ter mais influência na saída do sistema. Isso tem grande importância para evitar o fenômeno indesejado conhecido por *overfitting*. Dessa forma, aplicando um *dropout* de 80 % de chance de ocorrência, é esperado que a rede neural proposta consiga generalizar a classificação mesmo para entradas não treinadas. A relação do método de *dropout* com a redução do *overfitting* é apresentada e explicada no artigo de Srivastava *et al.*, (2014).

6.6 Resultados e discussões

O procedimento realizado para a avaliação do método proposto consiste basicamente nas etapas de aquisição de dados para o treinamento da rede neural, o treinamento em si e a etapa de validação com testes online durante a simulação para verificar a acurácia do método.

Os dados utilizados para treinamento e validação foram os mesmos 5 minutos/terreno coletados anteriormente para a confecção das Figuras 6.4 e 6.5 durante a caminhada do robô em cinco minutos de simulação para cada tipo de terreno. Estes cinco minutos geraram em torno de 800 perfis de impactos por terreno, sendo que 600 perfis foram selecionados aleatoriamente para o treinamento e os 200 restantes para a validação. Essa divisão entre dados de treinamento e de validação foi feita para que a avaliação da acurácia do modelo após a conclusão do seu treinamento pudesse ser feita para entradas não previstas durante o processo de aprendizagem, ou seja, os dados de avaliação contribuem para verificar o quanto o modelo final consegue ser generalista com relação às suas entradas. Esses valores foram transformados nas imagens que caracterizam o terreno e foram armazenados em um vetor de matrizes $\boldsymbol{X}$. A saída que foi chamada de $\boldsymbol{Y}$ é formada por vetores que indicam qual o terreno correspondente para cada entrada de $\boldsymbol{X}$. Por exemplo, para uma matriz de $\boldsymbol{X}$ que representa o impacto no terreno 2, a saída correspondente em $\boldsymbol{Y}$ é dada por: [0 1 0 0 0 0].

O treino foi configurado para ser executado no máximo com 200 épocas, que foi a quantidade necessária para chegar à conclusão que o modelo convergiu para uma acurácia máxima e que não apresentaria melhoras significativas mesmo com mais épocas (cada época corresponde à cada vez que todos os dados de treinamento passam pela rede para sua atualização), e a acurácia do modelo proposto foi avaliado para cada passo (*step*) do processo de aprendizagem, com os valores obtidos para validação. A acurácia ao longo do treinamento é apresentada na Fig. 6.12.

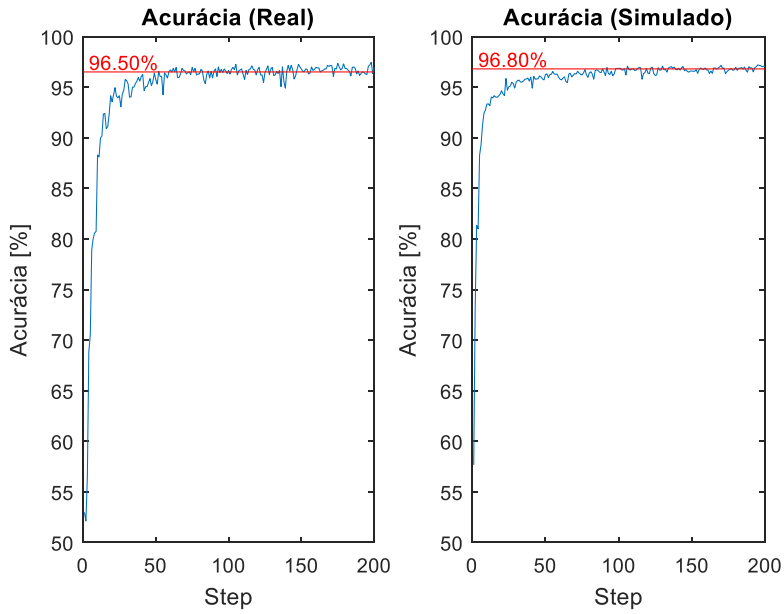


Figura 6.12 – Acurácia do modelo de classificação ao longo do processo de aprendizagem.

Observa-se que o valor médio máximo de acurácia para o conjunto de dados de validação com o modelo proposto atingiu o patamar de 96.50% para a situação de terrenos reais, e de 96.80% para identificação dos terrenos simulados, e não apresentou eminência de melhora após 200 épocas de treinamento. Pode-se dizer que em ambos casos, a classificação apresentou desempenho equivalente, e para ter uma ideia de quais terrenos foram melhores classificados, e quais apresentam ambiguidades na classificação, foram elaboradas as Tabelas 6.1 e 6.2 de porcentagem de classificação com base nos dados de validação. É importante notar que a porcentagem de acurácia mostradas nos gráficos da Fig. 6.12 descreve a classificação como um todo, ou seja, é a média das porcentagens individuais de acerto para cada tipo de terreno. Por isso, percebe-se que as Tabs. 6.1 e 6.2 possuem alguns valores de classificação inferiores aos obtidos nas figuras.

Tabela 6-1 – Relação entre os de entrada (na coluna da tabela), e as saídas percentuais obtidas para o conjunto de dados das amostras de validação no caso real.

	Edredom	Grama sin.	Borracha	Carpete	MDF	Cerâmica
Edredom	**97.0%**	2.5%	0.0%	0.0%	0.5%	0.0%
Grama sin.	0.5%	**95.5%**	1.0%	0.0%	1.0%	2.0%
Borracha	0.0%	2.5%	**94.0%**	2.0%	0.0%	1.5%
Carpete	0.0%	0.0%	0.5%	**97.0%**	2.0%	0.5%
MDF	0.0%	0.5%	0.0%	1.5%	**98.0%**	0.0%
Cerâmica	0.0%	1.0%	0.5%	0.0%	1.0%	**97.5%**

Tabela 6-2 - Relação entre os de entrada (na coluna da tabela), e as saídas percentuais obtidas para o conjunto de dados das amostras de validação no caso simulado.

	Terreno 1	Terreno 2	Terreno 3	Terreno 4	Terreno 5	Terreno 6
Terreno 1	**97.5%**	1.5%	1.0%	0.0%	0.0%	0.0%
Terreno 2	1.0%	**97.0%**	0.5%	0.5%	1.0%	0.0%
Terreno 3	1.0%	0.5%	**95.5%**	0.0%	2.0%	1.0%
Terreno 4	0.5%	0.0%	1.5%	**97.5%**	0.0%	0.5%
Terreno 5	0.5%	0.0%	2.0%	0.0%	**97.5%**	0%
Terreno 6	0.0%	0.0%	0.0%	1.5%	0%	**98.5%**

Com os resultados demonstrados nas Tabs. 6.1 e 6.2 percebe-se que todos os terrenos tiveram uma classificação igual ou superior à 94.0%, e para o cálculo de um intervalo de confiança desse resultado, seria necessária uma coleta de dados mais abrangente. Porém, com a quantidade de passos analisados, sendo 200 para cada situação,

percebe-se que o método proposto tem o potencial de classificação para diversos tipos de terrenos, mesmo que estes apresentem assinaturas de impactos semelhantes entre si, como foi observado na Figura 6.4 para os terrenos: grama sintética e edredom. Além disso, terrenos com características de pouco atrito para a superfície de contato do pé do robô, como constatou-se com a cerâmica e o MDF, mostraram clara distinção entre os mesmos, mesmo com ambos apresentando rigidez elevada em comparação com os demais.

Para os terrenos simulados, foi possível verificar que mesmo com a alteração da velocidade da caminhada durante os testes, foi possível realizar a classificação com desempenho equivalente aos resultados para os terrenos reais. Essa característica comprova que o método proposto tem a capacidade de generalizar os perfis de impacto.

É importante salientar que o tempo de processamento dos dados de impacto da rede neural proposta é de menos de 1 ms para um computador Intel® Core(TM) i7 2.6 GHz com 8 GB de memória RAM. Isso mostra o potencial de ser implementado de maneira online para *hardwares* mais simples.

O código utilizado para o treinamento e classificação estão no Apêndice E.

CAPÍTULO VII

CONCLUSÕES

Nesta dissertação foi feita uma revisão a respeito da caminhada bípede e os principais aspectos que envolvem esta área. O método LIPM foi apresentado e testado tanto em um robô real quanto no robô virtual, e algumas modificações no método foram apresentadas e justificadas para que esta pudesse ser ajustada para se adequar às limitações mecânicas do robô real, como as folgas e erros de posição no controle PID à nível de articulações. As modificações se mostraram funcionais para fazer com que o método fizesse o robô real se locomover por alguns instantes.

No ambiente de simulação, com a ausência das folgas mecânicas, verificou-se uma caminhada mais estável, com menos quedas e menos ajustes dos parâmetros adicionais aos originais do LIPM. Esse fato, permitiu confirmar a eficiência do método de geração de trajetória, porém, ao mesmo tempo, surgiu a necessidade de implementar algum mecanismo de estabilização da caminhada para pequenos distúrbios, principalmente para aqueles causados por mudanças nas características do terreno.

A mudança nos valores da constante K_p do controle PID se mostrou eficaz nessa tarefa de estabilização, e essa abordagem tem embasamento teórico e prático em trabalhos na área de robótica móvel com pernas, onde a alteração desta constante está relacionada com o controle de impedância, que tem o potencial de reduzir o efeito de impactos quando o robô interage com o ambiente. Foram executados testes no robô virtual para avaliar os efeitos da redução de K_p, e constatou-se que a redução de fato consegue melhorar a estabilidade geral durante a caminhada, mesmo nos momentos de transição entre terrenos. Porém, a redução excessiva da constante pode aumentar os erros de posição para cada articulação, o que causa instabilidades na caminhada. Esses erros de posição gerados com a redução de K_p se assemelha com o efeito de um filtro passa-baixa, que conforme foi mostrado, é possível ser reduzido com a utilização de uma compensação *feedforward*.

Com os parâmetros de caminhada e de constante K_p escolhidos para a caminhada

em seis diferentes tipos de terrenos foi possível ter um robô funcional para outro objetivo dessa dissertação, que é a identificação de terrenos durante a caminhada. Para essa etapa, foi feita uma revisão de técnicas de classificação utilizadas para diferentes tipos de robôs com pernas.

A busca por informações que caracterizam o terreno é geralmente o ponto de partida de todas as técnicas apresentadas, e a forma como esses dados são trabalhados determinam a acurácia e velocidade de aplicação do método. Nesta dissertação, os dados utilizados para o processo de identificação foram os valores de torque do joelho e do calcanhar, e os valores referentes à aceleração linear, velocidade angular, e posição angular fornecidas pelo IMU.

A determinação dos sensores usados para a classificação dos terrenos foi importante para a geração da chamada assinatura do terreno, onde foi gerado uma imagem no momento do impacto que pode ser visualmente distinguida para diferentes propriedades do piso.

A imagem gerada durante o impacto motivou o uso das redes neurais convolucionais, que apresenta características na sua estrutura e forma de operar que permite uma otimização computacional durante o processo de aprendizagem, e resultados positivos na classificação de imagens, já que consegue lidar até mesmo com atrasos na aquisição do sinal, sem prejudicar a classificação de forma significativa.

A rede neural utilizada é simples e com capacidade de processar as informações de entrada com rapidez e capacidade de ser implementada em tempo real. Os resultados demonstraram que o modelo adotado para a rede neural utilizada consegue manter uma taxa superior à 96 % para situações reais, o que mostra um desempenho comparável a técnicas aplicadas no estado da arte de identificação de terrenos.

7.1 Trabalhos futuros

Não foi feito uma análise da metodologia proposta de identificação de terrenos em momentos de transição, e isso pode ser feito futuramente para avaliação da eficácia do método nesse tipo de situação, e se são necessárias modificações, como por exemplo, utilizar a informação de dois passos, ao invés de apenas um, para a caraterização do terreno.

Para o caso da grama sintética, é interessante fazer testes para avaliar se a altura da grama consegue gerar características distintas para diferentes alturas, e conseguir dessa forma ser distinguida com a aplicação do método apresentado.

A quantidade de terrenos que pode ser distinguida durante a caminhada, provavelmente é afetada pela configuração da rede neural convolucional, e isso pode ser explorado em um trabalho futuro.

Além disso, é necessário avaliar em um robô real se a mudança da constante K_p tem a mesma eficiência em relação à estabilização como foi constatado no robô virtual. Os testes de caminhada com o robô real durante este trabalho mostraram dificuldades na estabilização mesmo para o teste em um só terreno.

Outro ponto importante que deve ser explorado é a aplicação de técnicas de estabilização do movimento em malha fechada e que leve em consideração o tipo de terreno em que o robô está pisando durante a caminhada.

REFERÊNCIAS BIBLIOGRÁFICAS

A Brief History of RoboCup. Disponível em: <http://www.robocup.org/a_brief_history_of_robocup>. Acesso em: 26 jul. 2017.

ABADI, M. *et al.* **TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems**. 2016. White paper.

ARAÚJO, M. A. **Aplicação do ROS no Controle de Movimento de Robôs Equipados com Motores Dynamixel em Linux de Tempo Real**. 2017. 114 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) - Universidade Federal de Uberlândia, Uberlândia.

ASTROM, K. J.; WITTENMARK, B. **Adaptive Control.** 2. ed. [S.l.]: Addison Wesley, 1995.

BEST, G. Terrain classification using a hexapod robot. In: AUSTRALASIAN CONFERENCE ON ROBOTICS AND AUTOMATION, 2013, **Proceedings**.

BOSWORTH, W. **Perception and control of robot legged locomotion over variable terrain**. 2016. 175 f. Doctor of Philosophy in Mechanical Engineering - Massachusetts Institute of Technology, Massachusetts.

BRAWNER, S. **SolidWorks to URDF Exporter**. Disponível em: <wiki.ros.org/sw_urdf_exporter>. Acesso em: 26 jul. 2018.

CAMPOLONGO, F.; CARIBONI, J.; SALTELLI, A. An effective screening design for sensitivity analysis of large models. **Environmental Modelling & Software**, v. 22, p. 1509-1518, jan. 2007.

CHIAVERINI, S.; MEDDAHI, A. A Null-Space based Behavioural control approach to coordinated motion of a humanoid robot. In: 2015 IEEE INTERNATIONAL CONFERENCE ON INFORMATION AND AUTOMATION, Aug. 2015, **Proceedings**. p. 20–25.

CIREŞAN, D. C. *et al.* Flexible, high performance convolutional neural networks for image classification. In: IJCAI INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2011, **Proceedings**. p. 1237–1242.

CLEVER, D. *et al.* A novel approach for the generation of complex humanoid walking sequences based on a combination of optimal control and learning of movement primitives.

Robotics and Autonomous Systems, v. 83, p. 287–298, 2016.

DAI, H.; TEDRAKE, R. Planning robust walking motion on uneven terrain via convex optimization. In: IEEE-RAS INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 2016, **Proceeedings**. p. 579–586.

DALEN, S.J. Van. **A Linear Inverted Pendulum Walk Implemented on TUlip**. 2012. 84 f. Master's thesis Universidad de Eindhoven, Eindhoven.

DYNAMICS, Boston. **Atlas The World's Most Dynamic Humanoid**. Disponível em: <https://www.bostondynamics.com/atlas>. Acesso em: 26 jul. 2018.

EIGEN. **No Title**. Disponível em: <http://eigen.tuxfamily.org/index.php?title=Main_Page>. Acesso em: 30 jul. 2018.

EINDHOVEN, UNIVERSITY OF TECHNOLOGY. **Zero-Moment Point Method for Stable Biped Walking**. Eindhoven, July 2009, 62p. Internship Report.

FALLON, M. F. Continuous humanoid locomotion over uneven terrain using stereo fusion. In: HUMANOID ROBOTS (HUMANOIDS), 2015 IEEE-RAS 15TH INTERNATIONAL CONFERENCE ON. IEEE, 2015. **Proceedings.**

FELIS, M. L. RBDL: an efficient rigid-body dynamics library using recursive algorithms. ***Autonomous Robots***, v. 41, n. 2, p. 495–511, 2017.

FILITCHKIN, P.; KATIE, B. Feature-based terrain classification for little dog. In: INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2012 IEEE/RSJ INTERNATIONAL CONFERENCE ON. IEEE, 2012. **Proceedings**.

GIGUERE, Philippe. **Unsupervised learning for mobile robot terrain classification**. 2010. 215 f. Doctor of Philosophy - School of Computer ScienceMcGill University, Montréal.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. ***Deep Learning***. [S.l.]: MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org >. Acesso em: 30 jul. 2018.

GUENTHER, M. **imu_tools**. Disponível em: < http://wiki.ros.org/imu_tools>. Acesso em: 26 jul. 2018.

GUIZZO, E. **Hubo II Humanoid Robot Is Lighter and Faster, Makes His Creator Proud.** Disponível em: < https://spectrum.ieee.org/automaton/robotics/humanoids/033010-hubo-ii-

humanoid-robot-is-lighter-and-faster> . Acesso em: 26 hul. 2018.

HOFFMANN, M.; STEPANOVA, K.; REINSTEIN, M. The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits. **Robotics and Autonomous System**, 2014.

HOGAN, N. Impedance Control: An Approach to Manipulation. AMERICAN CONTROL CONFERENCE, San Diego, CA, USA, 1984 , **Proceedings**. p. 304–313.

HONDANEWS. **Evolution of ASIMO: Advancements of Physical Capabilities**. Disponível em: <http://hondanews.com/channels/robotics-asimo/videos/asimo-new-capabilities-7>. Acesso em: 1 jan. 2018.

HONG, Y. D.; PARK, C. S.; KIM, J. H. Stable bipedal walking with a vertical center-of-mass motion by an evolutionary optimized central pattern generator. **IEEE Transactions on Industrial Electronics**, v. 61, n. 5, p. 2346–2355, 2014.

HROVATT, D. Survey of Advanced Suspension Developments and Related Optimal Control Applications. **Automatica**. v. 33, n.10, p. 1781-1817, 1997.

Humanoid Robot HRP-4. Disponível em: <http://global.kawada.jp/mechatronics/hrp4.html>. Acesso em: 23 jan. 2018.

KAJITA, S. *et al.* **Introduction to Humanoid Robotics**. [S.l: s.n.], 2014. v. 1.

KAJITA, S. *et al.* Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, v. 2, Oct. 2003. **Proceedings**. p. 1644–1650, 2003.

KAJITA, S. *et al.* The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation. IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, v. 1, n. 4, 2001. **Proceedings** p. 239–246.

KHADIV, M. *et al.* **Walking Control Based on Step Timing Adaptation**. *arXiv preprint arXiv:1704.01271v2* p. 1–14, 2017.

KINGMA, D. P; BA, J. L. **Adam: a method for stochastic optimization**. *arXiv preprint arXiv:1412.6980*, p. 1–15, 2014.

LECUN, Y. *et al.* Gradient-Based Learning Applied to Document Recognition. In: Proceedings of the IEEE, vol. 86, no. 11, Nov. 1998. **Proceedings**. p. 2278-2324. doi: 10.1109/5.726791.

LIU, Y. *et al.* Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3D-actuated Dual-SLIP model. *IEEE INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS*, Dec. 2015. **Proceedings**. p. 374–380.

MADGWICK, S.O.H. **An efficient orientation filter for inertial and inertial/magnetic sensor arrays**. p. 32, 2010.

MASON, S. *et al.* Balancing and walking using full dynamics LQR control with contact constraints. IEEE-RAS INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 2016. **Proceedings**. p. 63–68.

MISSURA, M. **Analytic and Learned Footstep Control for Robust Bipedal Walking**. 2016. Doctor of Philosophy - Universität Bonn, Bonn.

MISSURA, M.; BEHNKE, S. Self-stable Omnidirectional Walking with Compliant Joints. IEEE-RAS INTERNATIONAL CONFERENCE ON HUMANOID ROBOT, 2013. **Proceedings**.

MOORE, K. L. **Iterative Learning Control**. Iterative Learning Control for Deterministic Systems, p. 425–488, 1993.

MORADI, K.; FATHIAN, M.;GHIDARY, S. S. Omnidirectional walking using central pattern generator. **International Journal of Machine Learning and Cybernetics**, v. 7, n. 6, p. 1023–1033, 2016.

MOUSSA, S. **FUNDAMENTOS DE BIOMECANICA: APLICAÇOES DA MECATRONICA NO CORPO HUMANO**. [S.l.]: MOUSSA SALEN SIMHON, 2013.

NIELSEN, M. **Neural Networks and Deep Learning**. [S.l.]: Determination Press, 2015.

NISHIWAKI, K.; CHESTNUTT, J.; KAGAMI, S. Autonomous Navigation of a Humanoid Robot over Unknown Rough Terrain using a Laser Range Sensor. **The International Journal of Robotics Research**, v. 31, n. 11, p. 1251–1262, 2012.

ORDONEZ, C. Terrain identification for RHex-type robots. **SPIE Defense, Security, and Sensing. International Society for Optics and Photonic**, 2013.

PHIDGETS. **PhidgetSpatial 3/3/3 Basic**. Disponível em: <https://www.phidgets.com/?tier=3&catid=10&pcid=8&prodid=1025>. Acesso em: 30 jan. 2018.

POULAKAKIS, I.; GRIZZLE, J. W. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. ***IEEE Transactions on Automatic Control***, v. 54, n. 8, p. 1779–1793, 2009.

PRATT, J. E.; DILWORTH, P.; PRATT, G. A. Virtual Model Control of a Bipedal Walking Robot. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, vol. 1, 1997. **Proceedings**. p. 193–198.

ROBOCUP. **Objective**. Disponível em: <http://www.robocup.org/objective>. Acesso em: 26 jun. 2018.

ROBOTIS. **DYNAMIXEL**. Disponível em: <http://www.robotis.us/dynamixel/>. Acesso em: 23 jan. 2018.

ROBOTIS. **MX-106T / MX-106R Manual**. Disponível em: <http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-106.htm>. Acesso em: 26 jul. 2017.

SABUR, Mustafa. **joint_trajectory_controller**. Disponível em: <http://wiki.ros.org/joint_trajectory_controller>. Acesso em: 20 jan. 2018.

SAITO, Isaac. **rqt_reconfigure**. Disponível em: < http://wiki.ros.org/rqt_reconfigure>. Acesso em: 21 jan. 2018.

SCHWARZ, M.; BEHNKE, S. Compliant Robot Behavior using Servo Actuator Models identified by Iterative Learning Control. 17TH ROBOCUP INTERNATIONAL SYMPOSIUM, Eindhoven, Netherlands, 2013. **Proceedings**.

SICILIANO, B. *et al.* **Robotics: Modelling, Planning and Control**. [S.l: s.n.], 2008.

SRIVASTAVA, N. *et al.* Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014.

STEPHANIE, Annanda *et al.* **EDROM Humanoid Teen Size 2018**. Team Description Paper for Robocup Humanoid 2018.

TAKAYA, K. *et al.* Simulation environment for mobile robots testing using ROS and Gazebo. 20TH INTERNATIONAL CONFERENCE ON SYSTEM THEORY, CONTROL AND COMPUTING, ICSTCC 2016 - *JOINT CONFERENCE OF SINTES 20, SACCS 16, SIMSIS 20*, 2016. **Proceedings**. p. 96–101.

TANG, Y. TF.Learn: TensorFlow's High-level Module for Distributed Machine Learning. **CoRR**. vol. abs/1612.04251, 2016.

TONELLI, F. M. P.; RESENDE, R. R. VOLTANDO A ANDAR: O exoesqueleto robótico e o projeto Walk again em foco. **Nanocell News**. 1. 10.15729/nanocellnews, 2014.

VENÂNCIO, M. M. **Estudo e Simulação de Geração de Trajetórias do Caminhar de um Robô Humanoide Utilizando o Método do Pêndulo Invertido Linear**. 2016. 90 f. Trabalho de Conclusão de Curso, Universidade Federal de Uberlândia, Brasil.

VUKOBRATOVIĆ, M.; BOROVAC, B. Zero-Moment Point — Thirty Five Years of Its Life. **International Journal of Humanoid Robotics**, v. 01, n. 01, p. 157–173, 2004.

WALAS, K.; KANOULAS, D.; KRYCZKA, P. Terrain Classification and Locomotion Parameters Adaptation for Humanoid Robots Using Force / Torque Sensing. *IEEE-RAS INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS (HUMANOIDS 2016)*, 2016. **Proceedings**.

WESTERVELT, E. R. *et al.* **Feedback Control of Dynamic Bipedal Robot Locomotion**. [S.l.]: CRC Press, 2007.

WILSON, Matthew. **rqt_plot**. Disponível em: <http://wiki.ros.org/rqt_plot>. Acesso em: 20 jan. 2018.

YOUSUF, A. *et al.* Introducing kinematics with robot operating system (ROS). ASEE ANNUAL CONFERENCE AND EXPOSITION: MAKING VALUE FOR SOCIETY, n. 122, 2015. **Proceedings**.

YUAN, Q.; WANG, J. Design and experiment of the NAO Humanoid Robot's plantar tactile sensor for surface classification. INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND CONTROL ENGINEERING, n. 4, 2017. **Proceedings**. p. 931–935.

APÊNDICE A

Modelo do robô em URDF

```
<link
  name="TRUNK">
  <inertial>
    <origin
      xyz="-0.000142944293532541 -0.0159940135836359 0.119253638413231"
      rpy="0 0 0" />
    <mass
      value="1.4778155091901" />
    <inertia
      ixx="0.0125398154437975"
      ixy="0"
      ixz="0"
      iyy="0.016695640033345"
      iyz="0.000776177026664091"
      izz="0.01567882258573842" />
  </inertial>
</link>
<joint name="fixed" type="floating">
  <parent link="world"/>
  <child link="TRUNK"/>
  <origin xyz="0 0 0.50" rpy="0 0 0"/>
</joint>
<link
  name="HEAD0">
  <inertial>
    <origin
      xyz="0 -0.00045169 0.0273"
      rpy="0 0 0" />
    <mass
      value="0.072495" />
    <inertia ixx="1e-3" ixy="0" ixz="0" iyy="1e-3" iyz="0" izz="1e-3" />
  </inertial>
</link>
<link
  name="HEAD1">
  <inertial>
    <origin
      xyz="0.000492167176094648 0.0499545659310894 0"
      rpy="0 0 0" />
    <mass
      value="0.160467558648652" />
    <inertia ixx="1e-3" ixy="0" ixz="0" iyy="1e-3" iyz="0" izz="1e-3" />
  </inertial>
</link>
<joint
  name="J_HEAD0"
  type="continuous">
  <origin
    xyz="0 0.002 0.2785"
    rpy="0 0 -1.5708" />
  <parent
```

```
    link="TRUNK" />
  <child
    link="HEAD0" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
  <dynamics
      damping="0.7"
      friction="0.5"/>
</joint>
<joint
  name="J_HEAD1"
  type="continuous">
  <origin
    xyz="0 -0.0005 0.0425"
    rpy="1.5708 0 3.1416" />
  <parent
    link="HEAD0" />
  <child
    link="HEAD1" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
  <dynamics
      damping="0.7"
      friction="0.5"/>
</joint>
<link
  name="LARM0">
  <inertial>
    <origin
      xyz="0 0 0.0103025666414173"
      rpy="0 0 0" />
    <mass
      value="0.0066175226097181" />
    <inertia
      ixx="1.E-03"
      ixy="0"
      ixz="0"
      iyy="1.E-03"
      iyz="0"
      izz="1.E-03" />
  </inertial>
</link>
<joint
  name="J_LARM0"
  type="continuous">
  <origin
    xyz="-0.094 -0.0125 0.237"
    rpy="1.5708 3.14159 -1.5708" />
  <parent
    link="TRUNK" />
  <child
    link="LARM0" />
  <axis
    xyz="0 0 1" />
```

```
  <limit
    effort="10"
    velocity="8.58" />
  <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LARM1">
  <inertial>
    <origin
      xyz="-0.000203905909046856 -0.0358166170046635 -0.030724518771744"
      rpy="0 0 0" />
    <mass
      value="0.386889741732442" />
    <inertia
      ixx="0.001319462933467775"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.001386423060557297" />
  </inertial>
</link>
<joint
  name="J_LARM1"
  type="continuous">
  <origin
    xyz="-0.0308 0 0.036"
    rpy="3.1416 1.5708 0" />
  <parent
    link="LARM0" />
  <child
    link="LARM1" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LARM2">
  <inertial>
    <origin
      xyz="-0.000946892853796732 -0.0604743903550126 -0.030549305382329"
      rpy="0 0 0" />
    <mass
      value="0.417770884687889" />
    <inertia
      ixx="0.00130365609096481"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.00130522583526536" />
  </inertial>
</link>
```

```
<joint
  name="J_LARM2"
  type="continuous">
  <origin
    xyz="0.0307 -0.1235 -0.0308"
    rpy="0 1.5708 0" />
  <parent
    link="LARM1" />
  <child
    link="LARM2" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RARM0">
  <inertial>
    <origin
      xyz="0 0 0.0103025666414173"
      rpy="0 0 0" />
    <mass
      value="0.00661752260971814" />
    <inertia
      ixx="1.E-03"
      ixy="0"
      ixz="0"
      iyy="1.E-03"
      iyz="0"
      izz="1.E-03" />
  </inertial>
</link>
<joint
  name="J_RARM0"
  type="continuous">
  <origin
    xyz="0.094 -0.0125 0.237"
    rpy="-1.5708 0 -1.5708" />
  <parent
    link="TRUNK" />
  <child
    link="RARM0" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RARM1">
  <inertial>
    <origin
      xyz="-0.00020391 -0.035817 -0.030725"
```

```
      rpy="0 0 0" />
    <mass
      value="0.38689" />
    <inertia
      ixx="0.00131946"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.00138642" />
  </inertial>
</link>
<joint
  name="J_RARM1"
  type="continuous">
  <origin
    xyz="-0.0308 0 0.036"
    rpy="3.1416 1.5708 0" />
  <parent
    link="RARM0" />
  <child
    link="RARM1" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RARM2">
  <inertial>
    <origin
      xyz="-0.000946892853796774 -0.0604743903550127 -0.0305493053823289"
      rpy="0 0 0" />
    <mass
      value="0.417770884687889" />
    <inertia
      ixx="0.00130365609096481"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.00130522583526536" />
  </inertial>
</link>
<joint
  name="J_RARM2"
  type="continuous">
  <origin
    xyz="0.0307 -0.1235 -0.0308"
    rpy="0 1.5708 0" />
  <parent
    link="RARM1" />
  <child
    link="RARM2" />
  <axis
    xyz="0 0 -1" />
```

```
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LLEG0">
  <inertial>
    <origin
      xyz="0 0 0.012339611662812"
      rpy="0 0 0" />
    <mass
      value="0.137487098095279" />
    <inertia
      ixx="0.001"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
<joint
  name="J_LLEG0"
  type="continuous">
  <origin
    xyz="-0.057 -0.0325 -0.005"
    rpy="-3.14159265358979 0 -1.5707963267949" />
  <parent
    link="TRUNK" />
  <child
    link="LLEG0" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LLEG1">
  <inertial>
    <origin
      xyz="0 -0.014933566350856 -0.0542307430726147"
      rpy="0 0 0" />
    <mass
      value="0.484317524341268" />
    <inertia
      ixx="0.001258532465487086"
      ixy="0"
      ixz="0"
      iyy="0.001256538006237082"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
```

```
<joint
  name="J_LLEG1"
  type="continuous">
  <origin
    xyz="0.056 0 0.036"
    rpy="-1.5707963267949 0 -1.5707963267949" />
  <parent
    link="LLEG0" />
  <child
    link="LLEG1" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LLEG2">
  <inertial>
    <origin
      xyz="0.0896135033902749 -0.121669815792639 -0.0285739776731877"
      rpy="0 0 0" />
    <mass
      value="0.429581814867276" />
    <inertia
      ixx="0.001739443276879588"
      ixy="0.000469298592224479"
      ixz="0"
      iyy="0.00145781924000273"
      iyz="0"
      izz="0.00100793675092908" />
  </inertial>
</link>
<joint
  name="J_LLEG2"
  type="continuous">
  <origin
    xyz="0.0289999999999999 0 -0.0750000000000234"
    rpy="-1.5707963267949 0.78539816339745 -1.5707963267949" />
  <parent
    link="LLEG1" />
  <child
    link="LLEG2" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LLEG3">
  <inertial>
    <origin
      xyz="0.0938607247050214 0.021121469241447 -0.0266864244548486"
```

```
      rpy="0 0 0" />
    <mass
      value="0.253457988874213" />
    <inertia
      ixx="0.001157893827081503"
      ixy="0"
      ixz="0"
      iyy="0.001215254509125973"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
<joint
  name="J_LLEG3"
  type="continuous">
  <origin
    xyz="0.112429978208667 -0.169643181644254 0.00160000000000024"
    rpy="-0.0320557226319531 0.0283578707512148 -1.01721646459287" />
  <parent
    link="LLEG2" />
  <child
    link="LLEG3" />
  <axis
    xyz="-0.0283540701513945 -0.0320373469653184 0.999084408398647" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LLEG4">
  <inertial>
    <origin
      xyz="0.0207692569273853 -0.0149335663508559 -0.0309509783436903"
      rpy="0 0 0" />
    <mass
      value="0.484317524341267" />
    <inertia
      ixx="0.001"
      ixy="0"
      ixz="0"
      iyy="0.001256538006237082"
      iyz="0"
      izz="0.001258532465487085" />
  </inertial>
</link>
<joint
  name="J_LLEG4"
  type="continuous">
  <origin
    xyz="0.19759 0.040682 0.007913"
    rpy="0.034963 0.024684 -1.339" />
  <parent
    link="LLEG3" />
  <child
    link="LLEG4" />
  <axis
    xyz="0 0 -1" />
```

```
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="LFOOT">
  <inertial>
    <origin
      xyz="-0.00984398078216434 0.033081446163245 0.0742424926896279"
      rpy="0 0 0" />
    <mass
      value="0.474832547151362" />
    <inertia
      ixx="0.001"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.001" />
  </inertial>
  <collision>
    <origin
      xyz="-0.01 0.021 0.078"
      rpy="0 0 0" />
      <geometry>
        <box size="0.13 0.03 0.208"/>
      </geometry>
  </collision>
</link>
<joint
  name="J_LFOOT"
  type="continuous">
  <origin
    xyz="0.072 0 -0.031"
    rpy="0 -1.5708 0" />
  <parent
    link="LLEG4" />
  <child
    link="LFOOT" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RLEG0">
  <inertial>
    <origin
      xyz="0 0 0.012339611662812"
      rpy="0 0 0" />
    <mass
      value="0.137487098095279" />
    <inertia
```

```
      ixx="0.001"
      ixy="0"
      ixz="0"
      iyy="0.001"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
<joint
  name="J_RLEG0"
  type="continuous">
  <origin
    xyz="0.057 -0.0325 -0.005"
    rpy="-3.14159265358979 0 -1.5707963267949" />
  <parent
    link="TRUNK" />
  <child
    link="RLEG0" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RLEG1">
  <inertial>
    <origin
      xyz="0 -0.014934 -0.054231"
      rpy="0 0 0" />
    <mass
      value="0.48432" />
    <inertia
      ixx="0.00125853"
      ixy="0"
      ixz="0"
      iyy="0.00125654"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
<joint
  name="J_RLEG1"
  type="continuous">
  <origin
    xyz="0.056 0 0.036"
    rpy="-1.5707963267949 0 -1.5707963267949" />
  <parent
    link="RLEG0" />
  <child
    link="RLEG1" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
```

```
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RLEG2">
  <inertial>
    <origin
      xyz="0.089613503390275 -0.121669815792639 -0.0285739776731877"
      rpy="0 0 0" />
    <mass
      value="0.429581814867276" />
    <inertia
      ixx="0.001739443276879589"
      ixy="0.00046929859222448"
      ixz="0"
      iyy="0.001457819240002731"
      iyz="0"
      izz="0.00100793675092908" />
  </inertial>
</link>
<joint
  name="J_RLEG2"
  type="continuous">
  <origin
    xyz="0.0290000000000001 0 -0.075"
    rpy="-1.5707963267949 0.78539816339745 -1.5707963267949" />
  <parent
    link="RLEG1" />
  <child
    link="RLEG2" />
  <axis
    xyz="0 0 -1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RLEG3">
  <inertial>
    <origin
      xyz="0.093861 0.021121 -0.026686"
      rpy="0 0 0" />
    <mass
      value="0.25346" />
    <inertia
      ixx="0.00115789"
      ixy="0"
      ixz="0"
      iyy="0.00121525"
      iyz="0"
      izz="0.001" />
  </inertial>
</link>
<joint
  name="J_RLEG3"
  type="continuous">
  <origin
```

```
    xyz="0.112429978208668 -0.169643181644253 0.00160000000000014"
    rpy="-0.0320557226319534 0.0283578707512146 -1.01721646459286" />
  <parent
    link="RLEG2" />
  <child
    link="RLEG3" />
  <axis
    xyz="0.0283540701513948 0.0320373469653193 -0.999084408398647" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RLEG4">
  <inertial>
    <origin
      xyz="0.0207692569273853 -0.014933566350856 -0.0309509783436903"
      rpy="0 0 0" />
    <mass
      value="0.484317524341268" />
    <inertia
      ixx="0.001"
      ixy="0"
      ixz="0"
      iyy="0.001256538006237082"
      iyz="0"
      izz="0.001258532465487085" />
  </inertial>
</link>
<joint
  name="J_RLEG4"
  type="continuous">
  <origin
    xyz="0.19759 0.040682 0.007913"
    rpy="0.034963 0.024684 -1.339" />
  <parent
    link="RLEG3" />
  <child
    link="RLEG4" />
  <axis
    xyz="0 0 1" />
  <limit
    effort="10"
    velocity="8.58" />
      <dynamics
   damping="0.7"
   friction="0.5"/>
</joint>
<link
  name="RFOOT">
  <inertial>
    <origin
      xyz="0.009844 0.033081 0.074242"
      rpy="0 0 0" />
    <mass
      value="0.47483" />
    <inertia
```

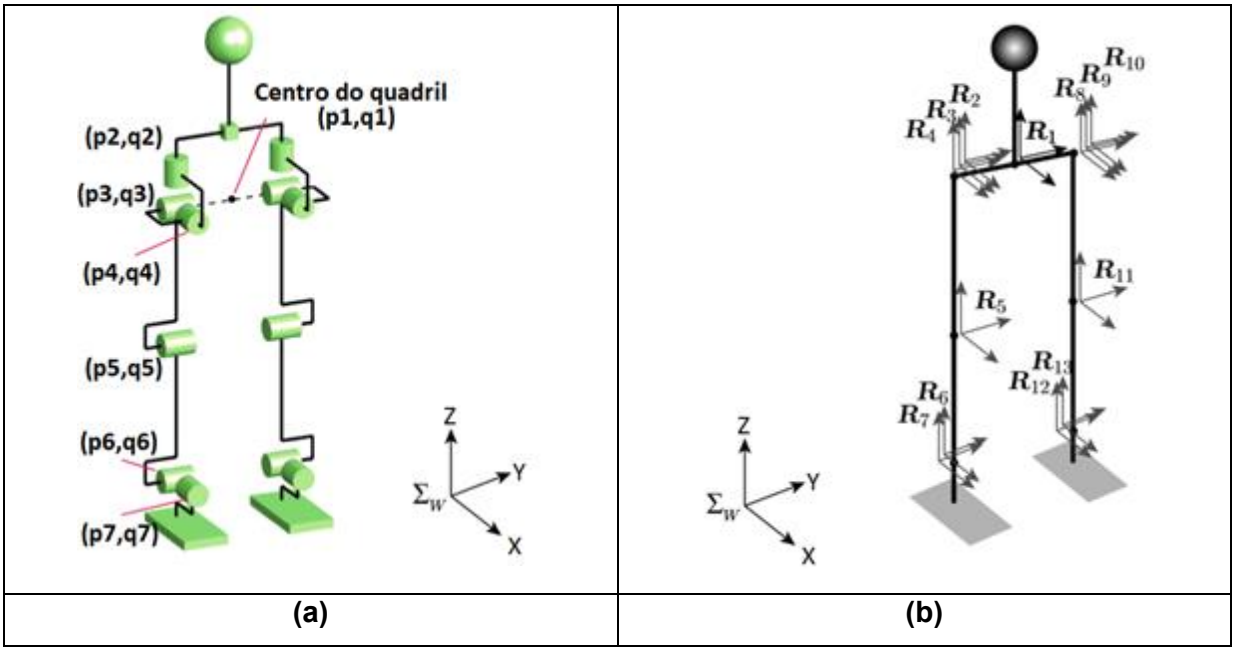
```
        ixx="0.001"
        ixy="0"
        ixz="0"
        iyy="0.001"
        iyz="0"
        izz="0.001" />
    </inertial>
    <collision>
      <origin
        xyz="0.01 0.021 0.078"
        rpy="0 0 0" />
        <geometry>
          <box size="0.13 0.03 0.208"/>
        </geometry>
    </collision>
  </link>
  <joint
    name="J_RFOOT"
    type="continuous">
    <origin
      xyz="0.072 0 -0.031"
      rpy="0 -1.5708 0" />
    <parent
      link="RLEG4" />
    <child
      link="RFOOT" />
    <axis
      xyz="0 0 -1" />
    <limit
      effort="10"
      velocity="8.58" />
        <dynamics
     damping="0.7"
     friction="0.5"/>
  </joint>
</robot>
```

APÊNDICE B

Cálculo da cinemática inversa e da posição do centro de massa

B.1 Cinemática inversa

O modelo cinemático inverso utilizado nessa dissertação foi o mesmo obtido por Kajita *et al.*, 2014, e será aqui demonstrado para o modelo de uma perna de seis graus de liberdade. A Figura B.1 (a) apresenta um esquema das posições de cada articulação e sua orientação tridimensional. Cada posição de articulação i em relação ao referencial fixo Σ_W, é representada pela variável $\boldsymbol{p_i}$ e a rotação relativa ao próprio eixo é dada por $\boldsymbol{q_i}$. Já a Figura B.1 (b) apresenta o posicionamento das orientações $\boldsymbol{R_i}$ de cada articulação em relação ao referencial fixo.

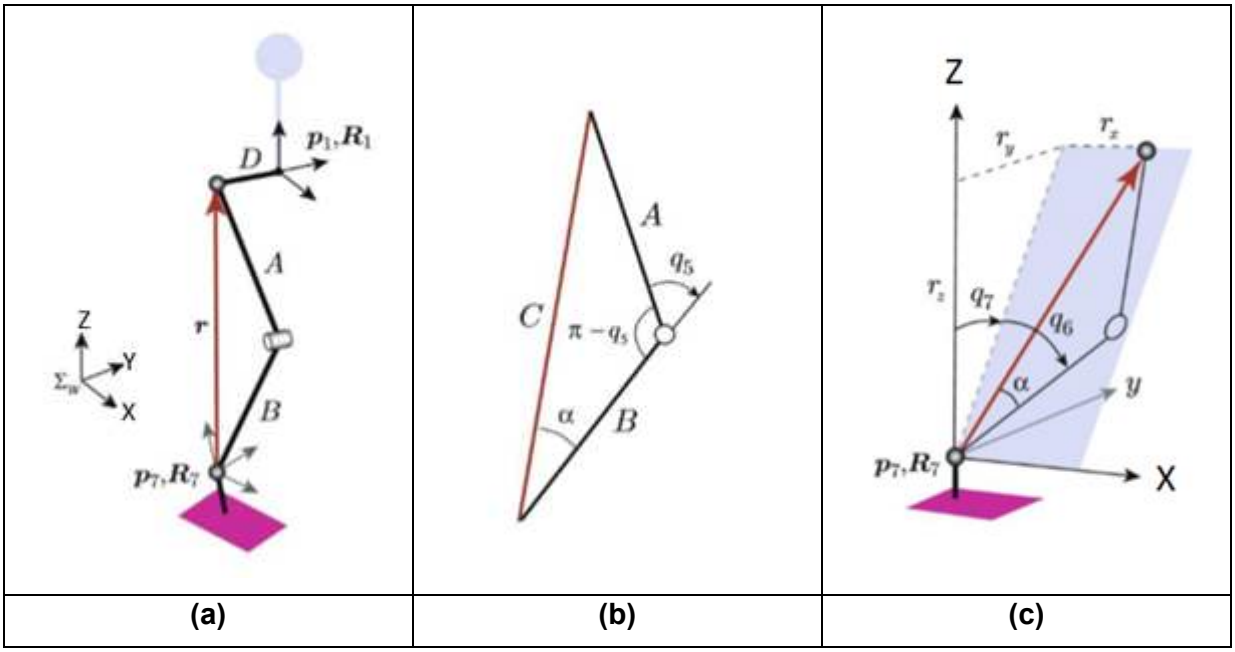


(a) **(b)**

Figura B.1 – (a) Modelo cinemático usado para o cálculo da cinemática inversa da perna direita com as respectivas nomenclaturas para cada articulação. (b) Representação das orientações $\boldsymbol{R_i}$ de cada articulação em relação ao referencial fixo Σ_W (Adaptado de Kajita *et al.*, 2014).

A Figura B.2 detalha as relações trigonométricas entre os ângulos das articulações e alguns ângulos auxiliares criados para a obtenção do modelo geométrico inverso. O

parâmetro A corresponde ao comprimento da parte superior da perna (seria o comprimento do osso fêmur no caso dos seres humanos. O parâmetro B é o comprimento da parte inferior da perna (ou o osso da tíbia), e o comprimento D é a distância entre a origem $\boldsymbol{p_1}$ e o primeiro servomotor de rotação da perna $\boldsymbol{p_2}$, ou seja, corresponde à distância entre o centro do quadril do robô com o início da perna.



(a)	(b)	(c)

Figura B.2 – Relações trigonométricas utilizadas para o cálculo da cinemática inversa. (a) Definições de localizações de variáveis utilizadas. (b) Ângulos relacionados à solução para o joelho q_5. (c) Ângulos relacionados à solução do tornozelo.

A cinemática inversa deve, portanto, levar em consideração os parâmetros constantes A, B e D, a posição e orientação do tronco para o instante analisado ($\boldsymbol{p_1}$,$\boldsymbol{R_1}$), e a posição e orientação desejada para os pés ($\boldsymbol{p_7}$,$\boldsymbol{R_7}$), Figura B.2 (a). Dessa forma, a posição do primeiro motor da perna, $\boldsymbol{p_2}$, é dada por:

$$\boldsymbol{p_2} = \boldsymbol{p_1} + \boldsymbol{R_1}\begin{bmatrix}0\\D\\0\end{bmatrix} \tag{B.1}$$

Em seguida, calcula-se a posição do ponto de referência, $\boldsymbol{p_1}$, para coordenadas locais no motor do tornozelo.

$$\boldsymbol{r} = \boldsymbol{R_7^T}(\boldsymbol{p_2} - \boldsymbol{p_7}) \equiv \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{B.2}$$

Através do vetor $\boldsymbol{r}$ pode-se encontrar a distância C entre o motor do tornozelo e o primeiro motor da perna:

$$C = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{B.3}$$

Utilizando a lei dos cossenos para o triângulo formado por ABC mostrado na Fig. B.2 (b), pode-se estabelecer a seguinte relação:

$$C^2 = A^2 + B^2 - 2ABcos(\pi - q_5) \tag{B.4}$$

Dessa maneira, o ângulo do motor do joelho, q_5, é dado por:

$$q_5 = -\cos^{-1}\left(\frac{A^2 + B^2 - C^2}{2AB}\right) + \pi \tag{B.5}$$

Aplicando a lei dos senos ainda no triângulo ABC, tem-se:

$$\frac{C}{\sin(\pi - q_5)} = \frac{A}{\sin(\alpha)} \tag{B.6}$$

E, portanto, o valor de α é igual a:

$$\alpha = \sin^{-1}\left(\frac{A\sin(\pi - q_5)}{C}\right) \tag{B.7}$$

Como mostrado na Fig. B.2 (c) é possível encontrar o valor dos ângulos do tornozelo q_7 e q_6 através das seguintes equações:

$$q_7 = atan2(r_y, r_z) \tag{B.8}$$

$$q_6 = -atan2\left(r_x, sign(r_z)\sqrt{r_y^2 + r_y^2}\right) - \alpha \tag{B.9}$$

A função $atan2(x, y)$ obtem o ângulo entre o vetor de coordenadas (x, y) e o eixo X.

Já a função $sign(x)$ retorna $+1$ se x não é negativo e -1 para o caso em que x é negativo.

As demais articulações podem ser obtidas através do rearranjo da seguinte relação que leva em consideração a rotação global de cada articulação:

$$\boldsymbol{R}_7 = \boldsymbol{R}_1\boldsymbol{R}_z(q_2)\boldsymbol{R}_x(q_3)\boldsymbol{R}_y(q_4)\boldsymbol{R}_y(q_5 + q_6)\boldsymbol{R}_x(q_7) \tag{B.10}$$

Rearranjando, tem-se:

$$\boldsymbol{R}_z(q_2)\boldsymbol{R}_x(q_3)\boldsymbol{R}_y(q_4) = \boldsymbol{R}_1^T\boldsymbol{R}_7\boldsymbol{R}_x(-q_7)\boldsymbol{R}_y(-q_5 - q_6) \tag{B.11}$$

Expandindo a relação B.11, obtém-se:

$$\begin{bmatrix} c_2c_4 - s_2s_3s_4 & -s_2c_3 & c_2s_4 + s_2s_3c_4 \\ s_2c_4 + c_2s_3s_4 & c_2c_3 & s_2s_4 - c_2s_3c_4 \\ -c_3s_4 & s_3 & c_3c_4 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{B.12}$$

Enfim, ao analisar, a equação B.12, e isolar corretamente os componentes, obtém-se os ângulos das articulações restantes:

$$q_2 = atan2(-R_{12}, R_{22}) \tag{B.13}$$

$$q_3 = atan2(R_{32}, -R_{12}s_2 + R_{22}c_2) \tag{B.14}$$

$$q_4 = atan2(-R_{31}, R_{33}) \tag{B.15}$$

Tendo como entrada os parâmetros relacionados às dimensões da perna A, B e D, a posição e orientação do quadril $\boldsymbol{p_1}$ e $\boldsymbol{q_1}$, e a posição e orientação do pé $\boldsymbol{p_7}$ e $\boldsymbol{R_7}$, é possível encontrar com as equações descritas, os valores dos seis graus de liberdade para os ângulos para toda a perna do robô: $[q_2 \ q_3 \ q_4 \ q_5 \ q_6 \ q_7]$.

Os valores do vetor $\boldsymbol{B}$ citado no Capítulo IV, que promove os efeitos na postura do robô indicados na Fig. 4.8, são utilizados nesse modelo geométrico inverso através da modificação das variáveis de entrada tanto do quadril ($\boldsymbol{p_1}$ e $\boldsymbol{q_1}$), quanto do pé ($\boldsymbol{p_7}$ e $\boldsymbol{R_7}$). Estas variáveis correspondem diretamente aos valores dos vetores $\widetilde{\boldsymbol{P}}$ e $\widetilde{\boldsymbol{Q}}$, que tem seus valores modificados pelos valores de $\boldsymbol{B}$ através das Eqs. 4.12-4.14. As equações seguintes mostram como ficam as entradas levando em consideração as correções posturais:

$$\boldsymbol{p_1} = \begin{bmatrix} \tilde{q}_x \\ \tilde{q}_y \\ \tilde{q}_z \end{bmatrix} \tag{B.16}$$

$$\boldsymbol{q_1} = \begin{bmatrix} \tilde{q}_\theta \\ \tilde{q}_\phi \\ \tilde{q}_\gamma \end{bmatrix} \tag{B.17}$$

$$\boldsymbol{p_7} = \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_y \\ \tilde{p}_z \end{bmatrix} \tag{B.17}$$

$$\boldsymbol{R_7} = \begin{bmatrix} \tilde{p}_\theta \\ \tilde{p}_\phi \\ \tilde{p}_\gamma \end{bmatrix} \tag{B.18}$$

B.2 Posição do centro de massa

A posição do centro de massa nessa dissertação foi determinada através de uma função específica da biblioteca RBDL (*Rigid Body Dynamics Library*) utilizada nos códigos implementados para geração de trajetórias. A função tem como entrada o modelo em URDF do robô apresentada no Apêndice A, e as posições angulares q_i de todas as articulações do robô humanoide, e tem como retorno a posição do centro de massa. O código utilizado para este cálculo é mostrado a seguir.

```
#include <rbdl/rbdl.h>
#ifndef RBDL_BUILD_ADDON_URDFREADER
#error "Error: RBDL addon URDFReader not enabled."
#endif
#include <rbdl/addons/urdfreader/urdfreader.h>

int main()
{
      //Carregando o arquivo URDF do robô
      urdfModel = RigidBodyDynamics::Model();
      if (!RigidBodyDynamics::Addons::URDFReadFromFile
("RoboHumanoide.urdf", &urdfModel, false))
      {
            std::cerr << "Error loading model urdf" <<std::endl;
            abort();
      }
      //Cria o vetor com três posições que vai receber a posição do
centro de massa
      RigidBodyDynamics::Math::Vector3d comPos
      //Declara o vetor com tordas as posições angulares em zero que
corresponde ao robô com a configuração ereta
      RigidBodyDynamics::Math::VectorNd roboPos =
RigidBodyDynamics::Math::VectorNd::Zeros(18);
      //Declara o vetor com tordas as velocidades angulares em zero
que corresponde ao robô parado
```

```
      RigidBodyDynamics::Math::VectorNd roboVel =
RigidBodyDynamics::Math::VectorNd::Zeros(18);
      //Função para cálculo do centro de massa
      RigidBodyDynamics::Utils::CalcCenterOfMass(urdfModel,qStateRBDL
.pos,qStateRBDL.vel,mass,comPos);
      //Exibe a posição obtida pela função anterior e enviada para a
variável comPos
      std::cout << "Posição do centro de massa: " << std::endl <<
comPos << std::endl;
      return 0;

}
```

Para mais detalhes a respeito da implementação desta função pode ser encontrado em:https://rbdl.bitbucket.io/df/d72/namespace_rigid_body_dynamics_1_1_utils.html#abfadeaeccca866d38e837e0a2cd9049d (Acessado em 18/08/2018).

APÊNDICE C

Método do Pêndulo Invertido Linear - LIPM

Neste Apêndice são detalhadas as equações e algoritmos utilizados para a obtenção da caminhada bípede utilizando LIPM. A demonstração das equações utilizadas neste apêndice podem ser consultadas em Kajita *et al.*, 2014.

O processo de cálculo contínuo das trajetórias do LIPM consiste, basicamente na, constante cálculo da trajetória do pé que está no ar, ao mesmo tempo que se calcula qual pé que está no chão. Após os dois pés chegarem nas suas posições finais, é feita a troca de qual pé deve estar no chão e qual deve estar no ar, e isso é feito continuamente. A orientação do robô no espaço tridimensional terá como referência o sistema de coordenadas mostrado na Fig. 4.1(b).

A geração das trajetórias será descrita por etapas:

C.1 Inicialização de variáveis que serão atualizadas ao longo do processo:

O tempo total atual desde o início da caminhada será denotado por T_{total}, e a variável que indica qual o n-ésimo passo que está sendo feito atualmente, será denotada por n. Dessa forma, inicializa-se as variáveis como:

$$T_{total} = 0 \tag{C.1}$$

$$n = 0 \tag{C.2}$$

As variáveis que descrevem a posição e a velocidade do centro de massa no espaço serão denotadas por $\boldsymbol{CM_{pos}}$ e $\boldsymbol{CM_{vel}}$, respectivamente. Para os vetores tridimensionais que informam a posição global dos pés direito e esquerdo será dada por: $\boldsymbol{P_{dir}}$ e $\boldsymbol{P_{esq}}$ respectivamente. Todas essas variáveis serão inicializadas tendo como referência a origem em $(0,0,0)$, a altura do centro de massa do robô dada por z_c, e as distância entre pernas que será chamada por $LegOpen$ como já foi explicado no Capítulo IV. Dessa forma, as variáveis para o valor de $n = 0$ ficam:

$$\boldsymbol{P_{dir}} = (0, -LegOpen, 0) \tag{C.3}$$

$$\boldsymbol{P_{esq}} = (0, LegOpen, 0) \tag{C.4}$$

$$\boldsymbol{CM_{pos}} = (0,0, z_c) \tag{C.5}$$

$$\boldsymbol{CM_{vel}} = (0,0,0)$$

As posições desejadas para os pés a cada momento da caminhada serão denotadas por $\boldsymbol{P_{dir}^{(D)}}$ e $\boldsymbol{P_{esq}^{(D)}}$, e serão inicializadas com os mesmos valores das variáveis $\boldsymbol{P_{dir}}$ e $\boldsymbol{P_{esq}}$ respectivamente.

C.2 Cálculo do vetor entre centro de massa e o pé de apoio:

Para a última posição e velocidade calculada para o centro de massa armazenados em $\boldsymbol{CM_{pos}}$ e $\boldsymbol{CM_{vel}}$ no passo anterior $n-1$, são obtidas as variáveis de posição e velocidade iniciais (x_i, y_i, $\dot{x}_i$ e $\dot{y}_i$) para o cálculo da trajetória atual do centro de massa ($x_{CM}(t)$, $y_{CM}(t)$, $\dot{x}_{CM}(t)$ e $\dot{y}_{CM}(t)$) no intervalo entre $[T_{total} \quad T_{total} + T_s]$. Além disso é levado em consideração a posição do pé de apoio atual dada por (x, y).

$$x_{CM}(t) = (x_i - x)\cosh\left(\frac{t}{T_c}\right) + T_c\dot{x}_i \sinh\left(\frac{t}{T_c}\right) \tag{C.6}$$

$$\dot{x}_{CM}(t) = \frac{(x_i - x)}{T_c}\sinh(t/T_c) + \dot{x}_i \cosh\left(\frac{t}{T_c}\right) \tag{C.7}$$

$$y_{CM}(t) = (y_i - y)\cosh\left(\frac{t}{T_c}\right) + T_c\dot{y}_i \sinh\left(\frac{t}{T_c}\right) \tag{C.8}$$

$$\dot{y}_{CM}(t) = \frac{(y_i - y)}{T_c}\sinh(t/T_c) + \dot{y}_i \cosh\left(\frac{t}{T_c}\right) \tag{C.9}$$

$$z_{CM}(t) = z_c \tag{C.10}$$

$$\dot{z}_{CM}(t) = 0 \tag{C.11}$$

$$T_c = \sqrt{\frac{z_c}{g}} \tag{C.12}$$

Caso o pé de apoio seja o pé direito, o cálculo que obtém o vetor $\boldsymbol{P_{CM-dir}}$ entre a posição do centro de massa do robô com o a posição do pé é dada por:

$$\boldsymbol{P_{CM-dir} = CM_{pos} - P_{dir}} \tag{C.13}$$

O mesmo pode ser feito para o pé esquerdo quando este for o pé de apoio.

No final dessa etapa, o tempo total deve ser atualizado para: $T_{total} = T_{total} + T_s$, assim como o valor do próximo passo: $n = n + 1$.

C.3 Cálculo da posição do próximo passo:

Para a atualização do próximo passo é calculado inicialmente a posição desejada $\boldsymbol{P}^{(\boldsymbol{D})}$ para o pé de apoio, de acordo com os valores desejados para o comprimento do passo em X e Y, definidos pelas variáveis S_x e S_y respectivamente. Estes valores são definidos de acordo com o valor de velocidade desejado para a caminhada. O valor de S_y para a caminhada em linha reta foi considerado constante e com o mesmo valor do parâmetro $LegOpen$. Portanto, para uma determinada velocidade de caminhada na direção X, definida por v_x, as equações para a determinação de S_x e S_y fica:

$$S_x = v_x T_s \tag{C.14}$$

$$S_y = LegOpen \tag{C.15}$$

Com essas variáveis é possível determinar a posição desejada $\boldsymbol{P}^{(\boldsymbol{D})}$ para cada passo n:

$$\boldsymbol{P}^{(\boldsymbol{D})}(n) = \begin{bmatrix} x^{(D)}(n) \\ y^{(D)}(n) \\ z^{(D)}(n) \end{bmatrix} = \boldsymbol{P}^{(\boldsymbol{D})}(n-1) + \begin{bmatrix} S_x \\ -(-1)^n S_y \\ 0 \end{bmatrix} \tag{C.16}$$

Agora deve-se calcular segmentos da caminhada $\bar{x}$ e $\bar{y}$, e velocidades $\overline{v_x}$ e $\overline{v_y}$ relativas às condições iniciais e finais do movimento do pêndulo:

$$\begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} \dfrac{S_x}{2} \\ \dfrac{(-1)^n S_y^{(n+1)}}{2} \end{bmatrix} \tag{C.17}$$

$$\begin{bmatrix} \overline{v_x} \\ \overline{v_y} \end{bmatrix} = \begin{bmatrix} \dfrac{C+1}{T_c S}\bar{x} \\ \dfrac{C-1}{T_c S}\bar{y} \end{bmatrix} \tag{C.18}$$

$$C = cosh(T_s/Tc) \quad S = senh(T_s/T_c) \tag{C.19}$$

Em seguida são calculados os estados desejados para o próximo passo tanto para a posição, representado por $\boldsymbol{E_{pos}}$ quanto para velocidade tanto para $\boldsymbol{E_{vel}}$:

$$\boldsymbol{E_{pos}} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} x^{(D)} + \bar{x} \\ y^{(D)} + \bar{y} \end{bmatrix} \tag{C.20}$$

$$\boldsymbol{E_{vel}} = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} \overline{v_x} \\ \overline{v_y} \end{bmatrix} \tag{C.21}$$

Finalmente, com os estados definidos por $\boldsymbol{E_{pos}}$ e $\boldsymbol{E_{vel}}$, pode-se calcular as posições do próximo passo do pé de apoio, que é dada por:

$$\boldsymbol{P} = -\frac{a(C-1)}{D}\left(\boldsymbol{E_{pos}} - \boldsymbol{CM_i} - T_c S \boldsymbol{\dot{CM}_i}\right) - \frac{bS}{T_c D}\left(\boldsymbol{E_{vel}} - \frac{S}{T_c}\boldsymbol{CM_i} - C\boldsymbol{\dot{CM}_i}\right) \tag{C.22}$$

$$D = a(C-1)^2 + b(S/T_c)^2 \tag{C.23}$$

As Eqs. (C.22) e (C.23) são obtidas através de uma minimização que tem como objetivo encontrar o melhor posicionamento $\boldsymbol{P}$ para o pé de apoio, de modo que o estado desejado $\boldsymbol{E_{pos}}$ e $\boldsymbol{E_{vel}}$ sejam atingidas tendo como condições iniciais o último valores do centro de massa identificados por $\boldsymbol{CM_i}$ e sua velocidade dada por $\boldsymbol{\dot{CM}_i}$. Os valores a e b são pesos no processo da minimização citada, e nessa dissertação eles foram escolhidos com os mesmos valores utilizados por Kajita *et al.* (2014), que foram $a = 10$ e $b = 1$.

Com o valor de $\boldsymbol{P}$ determinado, é feita a substituição no pé de apoio atual, sendo que se o último foi o pé direito, o novo pé de apoio será o esquerdo:

$$\boldsymbol{P_{esq}} = \boldsymbol{P} \tag{C.24}$$

C.4 Laço de repetição

O processo de geração contínua de caminhada consiste na inicialização das variáveis da caminhada descrita no tópico C.1, e depois disso a partir do passo do tópico C.2, o programa segue para o C.3, e depois retorna para o C.2 novamente. É importante ressaltar que as variáveis responsáveis por armazenar as posições dos pés direito e esquerdo devem ser alternadas constantemente para manter a alternância dos passos.

APÊNDICE D

Código em MATLAB para cálculo das variáveis utilizadas no controle feedforward

```
clc; clear all; close all;

file = load("RESULTADOS/Kp32Kd0.txt");


maxT   =  15;
dt     =  0.03;
nPts   = floor(maxT/dt) + 1;
nTotal = size(file)(1);
nIter  = nTotal/nPts;


t        = zeros(nPts,nIter);
qR  = qDotR = qd = qdDot = qM = qDotM = trjErr = U = err = t ;



for i = 1:nIter
      ini = (i-1)*nPts + 1;
      fim = ini + nPts - 1;

      t(:,i)          = file(ini:fim,1);
      tauR(:,i)       = file(ini:fim,2);
      qR(:,i)         = file(ini:fim,3);
      qDotR(:,i)      = file(ini:fim,4);
      qDot2R(:,i)     = file(ini:fim,5);
      qd(:,i)         = file(ini:fim,6);
      qdDot(:,i)      = file(ini:fim,7);
      qM(:,i)         = file(ini:fim,8);
      qDotM(:,i)      = file(ini:fim,9);
      Ubat(:,i)       = file(ini:fim,10);
      Curr(:,i)       = file(ini:fim,11);
      Temp(:,i)       = file(ini:fim,12);
      trjErr(:,i)     = file(ini:fim,13);
      U(:,i)          = file(ini:fim,14);
      err(:,i)        = file(ini:fim,15);

endfor


figure(1);
plot(err(1,2:end),'b*',"markersize",10);
title('Erro');
xlabel('i-th Iteracao');
ylabel('Erro Rms (rad)');

figure(2);
subplot(2,1,1);
plot(t(:,1),qd(:,1),'g',"linewidth",2);
hold on;
```

```
plot(t(:,1),qR(:,1),'b',"linewidth",2);
hold on;
plot(t(:,1),qM(:,1),'r',"linewidth",2);
title('Iter:0 Posicao');
xlabel('Tempo[s]');
ylabel('rad');
legend('Modificado','Desejado','Medido');
subplot(2,1,2);
plot(t(:,end),qd(:,end),'g',"linewidth",2);
hold on;
plot(t(:,end),qR(:,end),'b',"linewidth",2);
hold on;
plot(t(:,end),qM(:,end),'r',"linewidth",2);
title('Iter:END Posicao');
xlabel('Tempo[s]');
ylabel('rad');
legend('Modificado','Desejado','Medido');



figure(3);
subplot(2,1,1);
plot(t(:,1),qDotR(:,1),'b',"linewidth",2);
hold on;
plot(t(:,1),qDotM(:,1),'r',"linewidth",2);
hold on;
plot(t(:,1),qdDot(:,1),'g',"linewidth",2);
title('Iter:0 Velocidade');
xlabel('Tempo[s]');
ylabel('rad/s');
legend('Desejado','Medido','Modificado');
subplot(2,1,2);
plot(t(:,end),qDotR(:,end),'b',"linewidth",2);
hold on;
plot(t(:,end),qDotM(:,end),'r',"linewidth",2);
hold on;
plot(t(:,end),qdDot(:,end),'g',"linewidth",2);
title('Iter:END Velocidade');
xlabel('Tempo[s]');
ylabel('rad/s');
legend('Desejado','Medido','Modificado');

figure(4);
subplot(2,1,1);
plot(t(:,1),Ubat(:,1),'b');
hold on;
plot(t(:,1),Ubat(:,end),'r');
title('Voltagem');
xlabel('Tempo[s]');
ylabel('Volts[V]');
legend('Iter1','Iter2');
subplot(2,1,2);
plot(t(:,1),Curr(:,1),'b');
hold on;
plot(t(:,1),Curr(:,end),'r');
title('Corrente');
xlabel('Tempo[s]');
ylabel('Amperes[A]');
legend('Iter1','Iter2');
```

```
%Estimating Parameters

Z = (qM(:,1) - qR(:,1));
maxErrAntes = max(abs(Z));
maxErrAntes

Z = (qM(:,end) - qR(:,end));
maxErrDepois = max(abs(Z));
maxErrDepois

% Least Square Method for parameters estimation

sigma  = 1;
qDotS  = 0.1;
%Removing the first NAN term
t      = t(2:end,end);
tauR   = tauR(2:end,end);
qR     = qR(2:end,end);
qDotR  = qDotR(2:end,end);
qDot2R = qDot2R(2:end,end);
U      = U(2:end,end);
Ubat   = Ubat(2:end,end);

%Calculating Beta
beta   = exp(-abs(qDotR/qDotS).^sigma);

%Calculating the Matrix terms
phi0 = tauR;
phi1 = qDotR;
phi2 = sign(qDotR).*(1-beta);
phi3 = sign(qDotR).*(beta);


A = [phi0 phi1 phi2 phi3];

%Least Square method

alfa = ((inv(A'*A))*A')*U;

Carga = sum(abs(Curr(:,end)))
```

APÊNDICE E

Código em Python para o treinamento e identificação do terreno

```
import tensorflow as tf
import numpy as np
import tflearn
import rospy
import rosbag
import math
import random
from humanoid_msgs.msg import ImpactMsg
import matplotlib.pyplot as plt


def shuffleIO(X,Y):
    nInput = np.shape(X)[0]
    Xout   = np.zeros(np.shape(X))
    Yout   = np.zeros(np.shape(Y))
    randIn = np.arange(nInput)
    np.random.shuffle(randIn)

    for i in range(nInput):
        r = randIn[i]
        Xout[i,:] = X[r,:]
        Yout[i,:] = Y[r,:]
    return Xout,Yout

x0 = 0
def impact_callback(data):
    global x0
    if trained:
        x = get_msg(data)
        x = x.reshape([-1, dim1, dim2, 1])
        pred =  model.predict(x)
        print pred
        print np.argmax(pred)


def get_one_hot(targets, nb_classes):
    res = np.eye(nb_classes)[np.array(targets).reshape(-1)]
    return res.reshape(list(targets.shape)+[nb_classes])


def get_msg(msg):
    x =
np.concatenate((np.array(msg.torque3),np.array(msg.torque4),np.array(msg.gy
roX),np.array(msg.gyroY),np.array(msg.gyroZ),np.array(msg.accX),np.array(ms
g.accY),np.array(msg.accZ),np.array(msg.angX),np.array(msg.angY)))
    x = x.reshape((1,x.shape[0]))
    return x

fieldNames = ['edredrom','azulejo', 'campo' ,'mdf' ,'verso' ,'grama']

nTrain = 600
nFields = len(fieldNames)
```

```
bagFile = 'fieldsReal.bag'
bag = rosbag.Bag(bagFile)

for field in range(nFields):
    one_hot = get_one_hot(np.array([field]),nFields)
    count = 0
    for topic, msg, t in bag.read_messages(fieldNames[field]):
        #print msg
        x = get_msg(msg)

        if   count == 0:
            Xf  = x
            Yf  = one_hot
        else:
            Xf = np.concatenate((Xf,x))
            Yf = np.concatenate((Yf,one_hot))
        count += 1
    Xf,Yf = shuffleIO(Xf,Yf)
    if field == 0:
        X     = Xf[:nTrain]
        Y     = Yf[:nTrain]
        testX = Xf[nTrain:]
        testY = Yf[nTrain:]
    else:
        X     = np.vstack([X,Xf[:nTrain]])
        Y     = np.vstack([Y,Yf[:nTrain]])
        testX = np.vstack([testX,Xf[nTrain:]])
        testY = np.vstack([testY,Yf[nTrain:]])
bag.close()


X,Y  = shuffleIO(X,Y)
testX,testY = shuffleIO(testX,testY)


print np.shape(X)
print np.shape(Y)
print np.shape(testX)
print np.shape(testY)

input_dim  = np.shape(X)[1]
output_dim = np.shape(Y)[1]

trained = False

print input_dim
print output_dim

dim1 = 10
dim2 = 28

X = X.reshape([-1, dim1, dim2, 1])
testX = testX.reshape([-1, dim1, dim2, 1])



network = tflearn.input_data(shape=[None, dim1, dim2, 1], name='input')
network = tflearn.conv_2d(network, 16,3,
activation='relu',regularizer="L2")
network = tflearn.max_pool_2d(network, 2)
```

```
network = tflearn.local_response_normalization(network)
network = tflearn.conv_2d(network, 32, 3, activation='relu',
regularizer="L2")
network = tflearn.max_pool_2d(network, 2)
network = tflearn.local_response_normalization(network)

network = tflearn.fully_connected(network, 12, activation='tanh',
regularizer="L2")
network = tflearn.dropout(network, 0.8)
network = tflearn.fully_connected(network,output_dim, activation='softmax')
network = tflearn.regression(network, optimizer='adam',
learning_rate=0.001,loss='categorical_crossentropy', name='target')


"""
# Training
model = tflearn.DNN(network, tensorboard_verbose=0)
model.fit({'input': X}, {'target': Y}, n_epoch=200,
           validation_set=({'input': testX}, {'target': testY}),
snapshot_step=100,batch_size=64, show_metric=True,shuffle=True,
run_id='convnet_mnist')
model.save('modelReal.tflearn')
"""

#Load
model = tflearn.DNN(network, tensorboard_verbose=0)
model.load('modelReal.tflearn')

print np.shape(testY)

comp = np.zeros(shape = (np.shape(testX)[0],2),dtype=np.int)

print np.shape(comp)
for i in range(0,np.shape(testX)[0]):
    xi = testX[i,:,:,:]
    yi = testY[i,:]
    xi = xi.reshape([-1,dim1,dim2,1])
    pred  = model.predict(xi)
    comp[i,0] = str(np.argmax(yi))
    comp[i,1] = str(np.argmax(pred))
    s = 'Esperado: ' + str(comp[i,0]) + '  Obtido: ' + str(comp[i,1])
    print(s)

mat_comp =  np.zeros(shape = (nFields,nFields))
for i in range(0,np.shape(testX)[0]):
    x = comp[i,0]
    y = comp[i,1]
    s = 'X: ' + str(x) + '  Y: ' + str(y)
    print(s)
    mat_comp[x,y] = mat_comp[x,y] + 1
print mat_comp


"""
trained = True
rospy.init_node('humanoid_ddpg', anonymous=True)
rospy.Subscriber("humanoid_learning/impact", ImpactMsg, impact_callback)
rospy.spin()
"""
```