Pontifícia Universidade Católica
do Rio de Janeiro

Carlos Magno Catharino Olsson Valle

Controle por Torque Computado de um Robô Bípede Simulado com Locomoção via Aprendizado por Reforço.

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Ricardo Tanscheit
Co-Orientador: Prof. Leonardo Alfredo Forero Mendoza

Rio de Janeiro
Abril de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Carlos Magno Catharino Olsson Valle

Controle por Torque Computado de um Robô Bípede Simulado com Locomoção via Aprendizado por Reforço.

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Ricardo Tanscheit
Orientador
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Leonardo Alfredo Forero Mendoza
Co-Orientador
UERJ

Profa. Karla Tereza Figueiredo Leite
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Paulo Fernando Ferreira Rosa
IME

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro
Técnico Científico- PUC-Rio

Rio de Janeiro, 13 de abril de 2016

Carlos Magno Catharino Olsson Valle

Graduou-se em Engenharia de Controle e Automação pela Pontifícia Universidade Católica do Rio de Janeiro - Brasil em 2013.

Ficha Catalográfica

Valle, Carlos Magno Catharino Olsson

Controle por Torque Computado de um Robô Bípede Simulado com Locomoção via Aprendizado por Reforço / Carlos Magno Catharino Olsson Valle; orientador: Ricardo Tanscheit; co-orientador: Leonardo Alfredo Forero Mendoza – 2016.

90f.: il. (color.) ; 30 cm

Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2016.

Inclui referências bibliográficas.

1. Engenharia elétrica – Teses. 2. Humanoides. 3. Controle. 4. Aprendizado por reforço. I. Tanscheit, Ricardo. II. Mendoza, Leonardo Alfredo Forero. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Agradecimentos

Ao meu orientador, Professor Ricardo Tanscheit, e ao meu co-orientador, Professor Leonardo Mendoza, pelo apoio e parceria para a realização deste trabalho.

Ao CNPq e a PUC-Rio pelos auxilios concedidos.

Aos meus pais, Carlos Magno Olsson Valle e Maria Cristina da Silva Catharino, pela educação, atenção e carinho.

A todos os amigos da PUC-Rio.

A todos os amigos e familiares que de uma forma ou de outra me estimularam e me ajudaram.

Aos professores que participaram da banca examinadora.

Resumo

Valle, Carlos Magno Catharino Olsson; Tanscheit, Ricardo (Orientador); Mendoza, Leonardo Alfredo Forero (Co-Orientador). **Controle por Torque Computado de um Robô Bípede Simulado com Locomoção via Aprendizado por Reforço.** Rio de Janeiro, 2016. 90p. Dissertação de Mestrado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação apresenta o desenvolvimento de um controle híbrido de um robô do tipo humanoide Atlas em regime de locomoção estática para a frente. Nos experimentos faz-se uso do ambiente de simulação Gazebo, que permite uma modelagem precisa do robô. O sistema desenvolvido é composto pela modelagem da mecânica do robô, incluindo as equações da dinâmica que permitem o controle das juntas por torque computado, e pela determinação das posições que as juntas devem assumir. Isto é realizado por agentes que utilizam o algoritmo de aprendizado por reforço Q-Learning aproximado para planejar a locomoção do robô. A definição do espaço de estados, que compõe cada agente, difere da cartesiana tradicional e é baseada no conceito de pontos cardeais para estabelecer as direções a serem seguidas até o objetivo e para evitar obstáculos. Esta definição permite o uso de um ambiente simulado reduzido para treinamento, fornecendo aos agentes um conhecimento prévio à aplicação no ambiente real e facilitando, em consequência, a convergência para uma ação dita ótima em poucas iterações. Utilizam-se, no total, três agentes: um para controlar o deslocamento do centro de massa enquanto as duas pernas estão apoiadas ao chão, e outros dois para manter o centro de massa dentro de uma área de tolerância de cada um dos pés na situação em que o robô estiver apoiado com apenas um dos pés no chão. O controle híbrido foi também concebido para reduzir as chances de queda do robô durante a caminhada mediante o uso de uma série de restrições, tanto pelo aprendizado por reforço como pelo modelo da cinemática do robô. A abordagem proposta permite um treinamento eficiente em poucas iterações, produz bons resultados e assegura a integridade do robô.

Palavras-chave

Robôs humanoides; Q-Learning aproximado; Controle por troque computado; Robô Atlas; Simulador Gazebo; Controle hibrido.

Abstract

Valle, Carlos Magno Catharino Olsson. Tanscheit; Ricardo (Advisor). Mendoza; Leonardo Alfredo Forero (Co-Advisor). **Computed-Torque Control of a Simulated Bipedal Robot with Locomotion by Reinforcement Learning.** Rio de Janeiro, 2016. 90p. MSc Dissertation - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation presents the development of a hybrid control for an Atlas humanoid robot moving forward in a static locomotion regime. The Gazebo simulation environment used in the experiments allows a precise modeling of the robot. The developed system consists of the robot mechanics modeling, including dynamical equations that allow the control of joints by computed-torque and the determination of positions the joints should take. This is accomplished by agents that make use of the approximate Q-Learning reinforcement learning algorithm to plan the robot's locomotion. The definition of the state space that makes up each agent differs from the traditional cartesian one and is based on the concept of cardinal points to establish the directions to be followed to the goal and avoid obstacles. This allows the use of a reduced simulated environment for training, providing the agents with prior knowledge to the application in a real environment and facilitating, as a result, convergence to a so-called optimal action in few iterations. Three agents are used: one to control the center of mass displacement when the two legs are poised on the floor and other two for keeping the center of mass within a tolerance range of each of the legs when only one foot is on the ground. In order to reduce the chance of the robot falling down while walking the hybrid control employs a number of constraints, both in the reinforcement learning part and in the robot kinematics model. The proposed approach allows an effective training in few iterations, achieves good results and ensures the integrity of the robot.

Keywords

Humanoid Robots; Approximate Q-Learning; Computed-Torque Control; Atlas Robot; Gazebo Simulator; Hybrid Control.

Sumário

Lista de Figuras

1.
Introdução

A robótica industrial como é conhecida hoje teve origem em meados do século XX, logo após o desenvolvimento dos primeiros computadores digitais (1). Desde a sua primeira forma comercial, os braços robóticos (Figura 1) destinavam-se, entre outras funções, a executar operações que representassem um maior risco a saúde ou segurança aos operadores das linhas de produção.

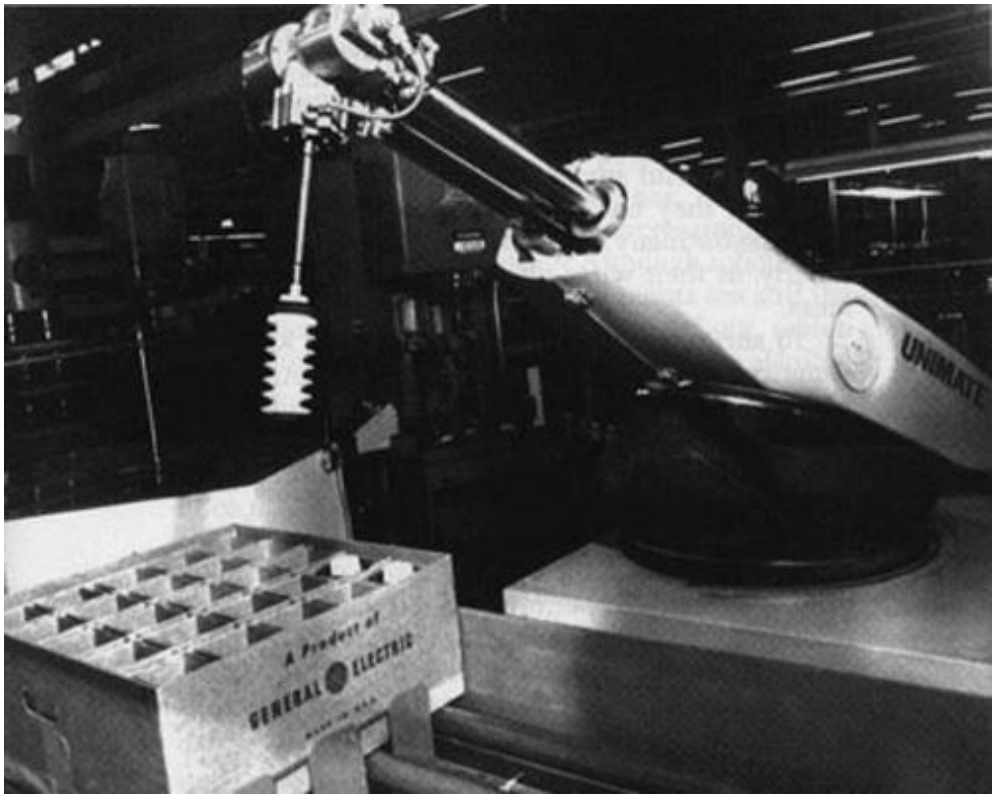

Figura 1 - UNIMATE, primeiro robô comercial na forma de um braço robótico (2).

Com o decorrer do tempo, os braços robóticos tornaram-se cada vez mais ágeis, precisos e capazes de operar maiores cargas (2). Com isto, houve um maior interesse da indústria em utilizar esta tecnologia, que proporcionava um maior controle sobre a incerteza na qualidade da produção (1). Assim, houve uma maior injeção de capital para pesquisas na área de robótica (1). Parte deste capital foi direcionada ao desenvolvimento de novas formas e aplicações, até que, na década

de 1970, surgiram as primeiras pesquisas envolvendo a formulação do deslocamento de bípedes (3, 4, 5).

Desde o princípio, os robôs bípedes foram utilizados para se estudar e aprimorar os métodos de controle utilizados para locomoção robótica. Por não possuírem uma base fixa ao chão, constituem-se em sistema subatuados (6), ou seja, nem todas as posições do robô apresentam estabilidade o suficiente para mantê-lo em pé.

1.1. O estado da arte em locomoção e estabilidade de robôs bípedes

Para que um robô seja capaz de realizar movimentos, faz-se necessário o desenvolvimento de um controle, pois sem este não há como mover de forma lógica as juntas do robô em prol de um movimento de locomoção ou simplesmente estabilização em uma posição especifica. Existem atualmente diversas metodologias para o desenvolvimento deste controle; as mais utilizadas baseiam-se na modelagem da dinâmica do robô e no cálculo do ciclo limite do movimento de caminhada (7). Outro método bastante empregado faz uso de reguladores quadráticos lineares (LQR) (8), oriundos da teoria de controle ótimo. Vale citar também a utilização da dinâmica inversa do robô para computar os torques necessários para posicionamento das juntas – em uma dada posição com determinada velocidade – e para a consequente geração de um movimento (9). Estes métodos são totalmente baseados na modelagem da dinâmica do robô, que é relativamente simples para robôs pequenos.

Para robôs mais complexos, a modelagem mecânica do robô, incluindo a sua dinâmica inversa, pode se tornar onerosa. Parcialmente por este motivo, empregam-se outros tipos de controle, como aqueles baseados em técnicas de inteligência computacional. As Redes Neurais do tipo *Spiking*, por exemplo, têm sido utilizadas para transcrever pulsos cerebrais biológicos em contrações musculares de um músculo artificial em um braço robótico (10). Utilizam-se também Redes Neurais do tipo clássico para controlar e realizar movimentos mais similares aos de seres humanos com os dedos de uma mão robótica (11) ou para convergir em controles puramente neurais capazes de administrar a força necessária para realizar uma tarefa sem conhecimento prévio das dinâmicas

envolvidas (12). A Lógica *Fuzzy* tem sido empregada no controle de guinada nos tornozelos do robô humanoide, possibilitando a redução do gasto energético utilizado pela estabilidade bípede (13). O Aprendizado por Reforço pode ser usado para determinar as atuações pneumáticas necessárias para manter um robô humanoide dinamicamente estável (14) ou para refinar a execução de uma tarefa específica, como, por exemplo, empurrar caixas (15). Tais métodos possibilitam simplificar ou até substituir a modelagem mecânica, mas, quanto mais reduzida esta for, maior será o esforço sobre o método computacional. Isto pode ocasionar uma alta demanda computacional, seja por uma maior necessidade de simulações ou por envolver um grande número de variáveis. Para contornar este problema, faz-se uso de modelos híbridos, que empregam conceitos de mecânica e de inteligência computacional. Exemplificando, pode-se utilizar o modelo da mecânica para controlar as posições e manter a estabilidade do robô e métodos de apoio à decisão para decidir que ação executar (16, 17), que trajetória deve ser percorrida (18), como realizar uma trajetória com baixo consumo energético do robô (19) ou até como cooperar com outros robôs para uma tarefa em equipe (20, 21). Os modelos híbridos possibilitam a geração de soluções mais especializadas e, portanto, mais poderosas, principalmente para sistemas com elevados graus de liberdade como o abordado nesta dissertação.

1.2. Motivação

Em março de 2011, um terremoto seguido de um tsunami, que devastou boa parte do Japão, ocasionou uma falha crítica nos equipamentos de segurança da usina nuclear de *Fukushima Daiichi* (22). Esta falha permitiu o vazamento de radiação para algumas áreas operacionais da usina e, se nenhuma medida fosse tomada para aliviar a pressão interna dos reatores, poderia acarretar em uma contaminação ambiental de nível global.

Com o objetivo de avaliar os danos e tentar reparar o que fosse possível (23), foram utilizados diversos tipos de robôs para acessar áreas com altos níveis de radiação. Apesar dos robôs terem desempenhado um bom trabalho eles não eram habilitados a operar, de forma rápida, precisa e algumas vezes autônoma, no ambiente do desastre de modo que ainda foi necessário expor a altos níveis de

radiação mão de obra humana. Por isso ficou claro que os robôs e as técnicas de controle desenvolvidas até o momento eram pouco hábeis a utilizar equipamentos e instalações desenvolvidas para humanos, como válvulas, maçanetas e escadas. Este incidente veio a ressaltar a real utilidade dos robôs bípedes, como força de resposta a desastres (24).

Em 2013, a Agência de Projetos de Pesquisa Avançada de Defesa dos Estados Unidos (DARPA) lançou um desafio destinado a acelerar o desenvolvimento tecnológico de robôs bípedes. Este desafio impulsionou o desenvolvimento e o compartilhamento de interfaces de simulação mais realistas, bem como o desenvolvimento de modelos robóticos com mais sensores e de maior precisão (25).

Os robôs bípedes, especialmente os mais modernos, são sistemas dinâmicos subatuados (6) e, assim, requerem sistemas de controle demasiadamente complexos. Para sistemas subatuados de baixa complexidade, é possível desenvolver controles que usam a modelagem da dinâmica do sistema a ser controlado, como, por exemplo, o cálculo da dinâmica inversa (9). Contudo, no caso de sistemas de maior complexidade, esta abordagem pode se tornar praticamente inviável. Por isso, como mencionado anteriormente, é vantajoso o desenvolvimento de modelos híbridos que utilizem parte dos modelos da mecânica para controlar posição e velocidade e métodos de inteligência computacional para tomadas de decisão ou planejamento de locomoção.

1.3. Objetivos

O objetivo dessa dissertação é desenvolver um controle do tipo hibrido capaz de locomover um robô humanoide bípede virtual. Em relação à mecânica, serão modeladas a dinâmica e as atuações necessárias para manter um equilíbrio estático, e serão calculadas as regiões de estabilidade estática do robô humanoide, No que diz respeito à inteligência, um controle baseado em aprendizado por reforço será utilizado para planejar e executar a locomoção, procurando assegurar a integridade estrutural do robô, ou seja, garantindo que o sistema de aprendizado por reforço não experimente posições de juntas que possam representar

instabilidade. Desta forma, o sistema de controle poderá ser aplicado a um robô real sem oferecer risco de queda na fase inicial do aprendizado.

1.4. Estrutura da dissertação

Esta dissertação será dividida em mais cinco capítulos, cada um abordando uma área de conhecimento teórico ou prático utilizado no desenvolvimento do controle proposto. No segundo capítulo são apresentados o modelo de robô bípede utilizado bem como o ambiente de simulação em que o controle foi desenvolvido. O terceiro capítulo apresenta o desenvolvimento do modelo da dinâmica dos membros do robô, o controle de torque computado e a definição de posições estáveis para o aprendizado por reforço. No quarto capítulo serão evidenciados os fundamentos do algoritmo de aprendizado por reforço utilizado e as definições das variáveis que modelam o problema e a forma como o aprendizado é adquirido. O quinto capítulo apresenta os resultados obtidos e o sexto capítulo conclui o trabalho e sugere possíveis desdobramentos desta pesquisa.

2.
Modelo robótico e ambiente de simulação

Este capítulo tem por objetivo apresentar o modelo de robô humanoide e o ambiente de simulação escolhidos para a implementação do controle proposto.

2.1.
Robô humanoide Atlas

Escolheu-se o modelo robótico Atlas (Figura 2) por ser este o mais recentemente desenvolvido para tarefas de busca e resgate. O robô Atlas é um robô humanoide bípede desenvolvido pela empresa americana de robótica *Boston Dynamics*, com financiamento e supervisão da DARPA. É construído em alumínio e titânio de aviação e pesa aproximadamente 150 quilos (26, 27).

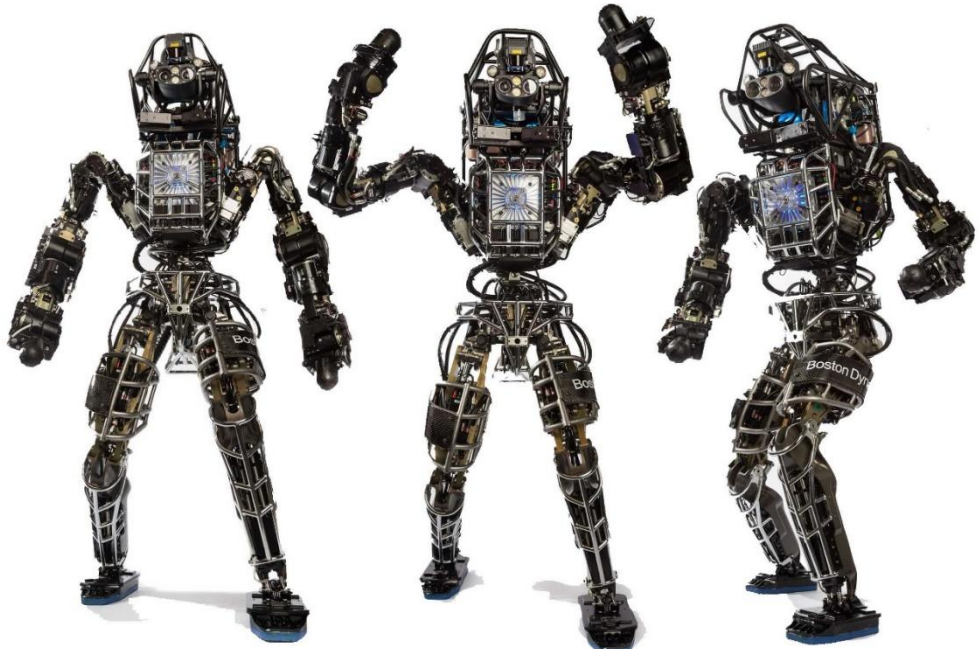

Figura 2 - Robô humanoide Atlas em três posições distintas (26).

O modelo é composto de dois braços, duas pernas, um torço e uma cabeça, agregando um total de 28 juntas de atuação hidráulica. A Figura 3 apresenta a posição de cada junta e os graus de liberdade.

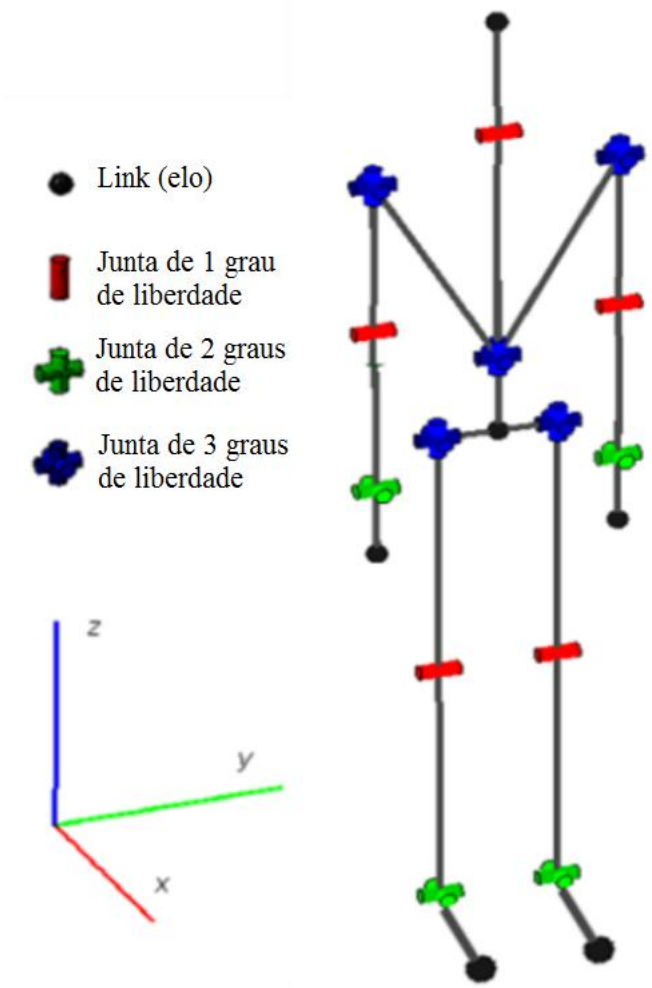

Figura 3 - Esquemático das juntas do modelo robótico.

Cada junta contém um atuador hidráulico, que nesta aplicação tem uma funcionalidade similar à de um motor, e sensores que determinam seu ângulo e sua velocidade. Tal sensoriamento é essencial para que a posição do atuador seja determinada. Cada junta é controlada por um controle PID, que determina o torque aplicado. Como as juntas são conectadas entre si, existem efeitos dinâmicos que precisam ser compensados. Tais compensações e malhas de controle serão abordadas no Capítulo 3.

Na região da pélvis, há uma central inercial de medição (IMU – *Inertial Measurement Unit*) que, utilizando acelerômetros, magnetômetros e giroscópios, consegue fornecer dados relativos à variação angular da base de referência. Isto também será discutido no Capítulo 3.

Nas extremidades dos braços e pernas existem sensores de pressão e torque, que fornecem dados importantes para a compensação das forças externas ao robô e obtenção de equilíbrio, o qual será explicado no Capítulo 3.

O robô também é equipado, na cabeça, com um *laser rangefinder*, que serve para medir as distâncias aos objetos, e com câmeras estéreo. Isto possibilita o uso de técnicas de visão computacional para mapeamento do ambiente em que o robô está inserido. Nesta dissertação tais sensores não foram utilizados, pois o objetivo era desenvolver um algoritmo de controle e locomoção em um ambiente plano.

O robô Atlas foi projetado para navegar em terrenos acidentados e escalar obstáculos usando seus braços e pernas de forma independente, embora a versão 2013 do protótipo, na qual este trabalho se baseia, fosse acoplada a uma fonte de alimentação externa para manter a estabilidade.

Com o modelo robótico selecionado, o próximo passo foi escolher um simulador capaz de retratar de forma realista o modelo em si e a sua interação com o ambiente.

2.2. Ambiente de simulação Gazebo

O objetivo fundamental da utilização de ambientes de simulação é o desenvolvimento de um agente virtual capaz de pensar e agir de acordo com a sua percepção do mundo. Desta forma, o conhecimento adquirido poderá ser transferido para o robô real. Os ambientes de simulação devem ser robustos e precisos, emulando o mundo real e o robô da melhor forma possível, permitindo uma melhor portabilidade entre o código do robô simulado e o do robô real.

Testar e desenvolver qualquer algoritmo de aprendizado – que necessita de várias iterações – diretamente num robô real pode trazer prejuízos significativos. As plataformas robóticas, quer sejam os mais simples braços articulados ou os mais complexos robôs humanoides, são usualmente muito caras. A utilização de ambientes de simulação para pesquisa, desenvolvimento e teste na robótica possui várias vantagens frente à utilização de robôs reais (28). As principais são:

- Menor custo;
- Facilidade de desenvolvimento e teste de novos modelos de robôs;
- Facilidade de teste de novos algoritmos;
- Diminuição de tempo de desenvolvimento e teste;
- Todos os testes podem ser feitos sem danificar o robô real;

- Para otimização e testes repetitivos, a utilização de um modelo virtual reduz substancialmente o tempo entre cada iteração.

Como os simuladores físicos contêm algumas simplificações na modelagem do mundo, os resultados da simulação e os do mundo real poderão diferir. Este problema é denominado *reality gap* no campo da robótica evolutiva (29). Isto significa que os modelos de softwares desenvolvidos em simulação necessitam de pequenos ajustes para utilização em ambiente real. O volume de alterações necessárias é inversamente proporcional à precisão da simulação.

Para selecionar um ambiente de simulação foram levados em consideração basicamente três itens essenciais:

- Acesso a simuladores de alto desempenho de física tridimensional para cálculo da dinâmica do modelo, do ambiente e de interações entre ambos: tal requisito é necessário para que o movimento efetuado no simulador seja similar ao realizado em um ambiente real;
- Aquisição de sensoriamento virtual com possibilidade de adição de ruídos: tal requisito é necessário para simular uma aquisição realista de dados, que, em geral, podem ser afetados por ruído;
- Modelo do robô Atlas já modelado para o ambiente: modelar o robô sem acesso aos parâmetros específicos de fabricação seria impraticável.

Como o modelo robótico escolhido era consideravelmente novo, apenas o simulador Gazebo (Figura 4) satisfazia a tais requisitos:

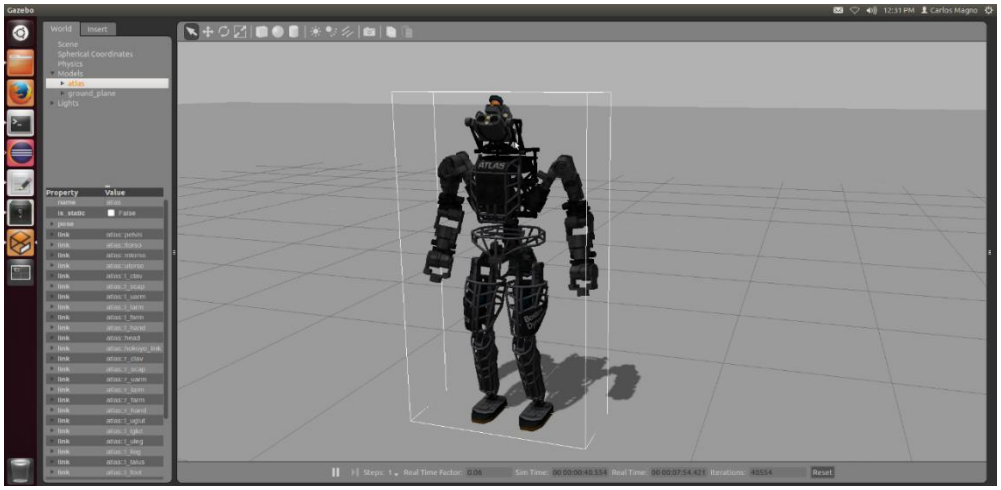

Figura 4 - Demonstração do simulador Gazebo em uso.

O Gazebo é uma solução desenvolvida pela *Open Source Robotics Foundation* (OSRF), uma organização sem fins lucrativos que fornece programas de código aberto para auxilio a pesquisas no campo da robótica (30). O projeto de código aberto usado para ambientes UNIX fornece uma interface limpa e simples dos sensores e atuadores do robô através de uma rede IP. O Gazebo fornece ambientes de simulação 3D e é capaz de simular uma população de robôs a interagir com o ambiente. Além disso, permite a simulação das propriedades físicas dos objetos simulados, o que permite uma melhor aproximação a um modelo real.

É possível dividir o processo de simulação em três componentes fundamentais: simulador (Gazebo), *framework*, e programa de controle.

Um *framework* é uma espécie de gestor da simulação, em que os dados provenientes das bibliotecas que simulam os sensores, no simulador, podem ser direcionados aos programas que processam estes dados e determinam a melhor ação a ser tomada. Tal ação é então direcionada ao simulador, que a executa, e isto se repete até a interrupção da simulação. A necessidade de um *framework* justifica-se frente à grande quantidade de linguagens em que o algoritmo de controle pode ser desenvolvido. Os desenvolvedores do Gazebo fornecem uma solução própria de simulação chamada Player. Entretanto, como o projeto pretendia usar o modelo de robô humanoide Atlas – utilizado por diversas equipes que participaram do desafio de robótica da DARPA (25) –, fez-se uso do *framework* ROS (*Robot Operating System*), com código aberto (31) e desenvolvido especificamente para aquele desafio.

No início deste trabalho, as ferramentas só estavam disponíveis para o sistema operacional Linux, o que fez com que se utilizasse a distribuição Ubuntu na versão 12.04.4 (denominada *precise*), a mais estável na ocasião. O programa de controle foi desenvolvido na plataforma Eclipse, versão Luna, na linguagem C++.

No próximo capítulo serão abordados os conceitos necessários para o desenvolvimento dos modelos da dinâmica em que o controle proposto se apoia.

3.
Conceitos básicos para o desenvolvimento do controle

Este capítulo aborda os conceitos básicos necessários para modelagem da dinâmica do robô e um controle capaz de compensar possíveis problemas gerados pela dinâmica do sistema.

3.1.
Anatomia de um robô

Um robô humanoide pode ser concebido como uma serie de manipuladores conectados entre si (32). Cada extremidade (braços, cabeça e pernas) e o torso são tratados como manipuladores individuais.

Um manipulador consiste em uma série de corpos, rígidos ou não, conectados entre si por juntas. Cada corpo envolvido no manipulador é chamado de *link* ou elo e a combinação dos elos e juntas é denominada *linkage* ou cadeia (Figura 5). O último ou o mais afastado *link*, em relação à base, é chamado de *end effector* ou extremidade atuante, pois em geral é neste *link* em que uma garra ou ferramenta (no caso de um braço) é acoplada.

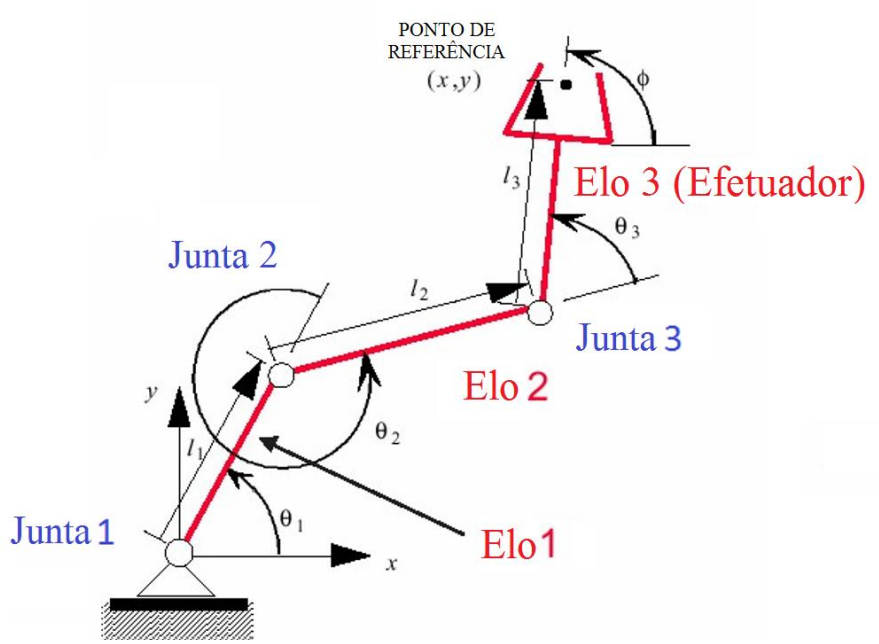

Figura 5 - Exemplo de manipulador bidimensional.

Com o intuito de facilitar a simulação serão considerados os corpos que compõem o manipulador como rígidos, ignorando elasticidade e qualquer deformação causada por condições de elevado estresse estrutural.

Existem dois tipos básicos de conexões entre um par de *links*, como mostrado na Figura 6. O primeiro é uma junta do tipo prismática onde um par de *links* faz um deslocamento translacional ao longo de um eixo fixo. Em outras palavras, um *link* desliza sobre o outro ao longo de uma linha reta. Assim, esta junta também é chamada de junta deslizante. O segundo tipo de junta é o de revolução, em que um par de *links* gira ao redor de um eixo fixo. Tal junta é muitas vezes denominada dobradiça, articulada ou de rotação.

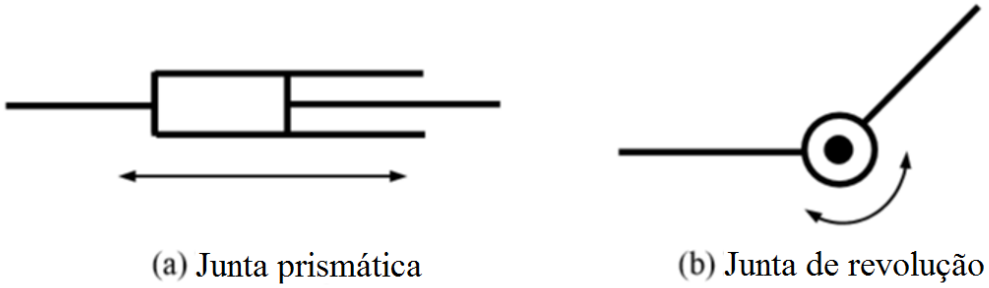

Figura 6 - Tipos básicos de juntas (32).

Em um sistema robótico, cada junta que se mova em uma única direção independente representa um grau de liberdade. Em um ambiente tridimensional é necessário um mínimo de seis graus de liberdade (seis juntas distintas abrangendo os três eixos de coordenadas) para que a extremidade do sistema se mova livremente em diversas orientações para uma mesma posição do *end effector*. Qualquer número superior a seis representa uma redundância. O robô humanoide Atlas possui 28 juntas, o que representa 28 graus de liberdade. Se o robô for tratado como uma cadeia única, o controle das juntas será altamente complexo. Por isso pretende-se determinar uma serie de cadeias menores que operem de forma coordenada, tornando menos complexos os modelos de cada cadeia e, consequentemente, menos onerosos computacionalmente.

3.2. Cinemática

A cinemática é um ramo da mecânica que descreve o movimento de pontos, corpos e sistemas de corpos. Para isso utiliza-se o estudo da trajetória de pontos,

linhas, outros objetos geométricos (como eixos de coordenadas, neste caso) e suas propriedades diferenciais como velocidade e aceleração (33).

Como uma tarefa especifica é feita pelo movimento conjunto de toda a cadeia do manipulador, a cinemática é fundamental para o controle. Nesta seção serão demonstradas as ferramentas matemáticas necessárias para o desenvolvimento das equações que descrevem o movimento da cadeia.

3.2.1. Posição e orientação de corpos rígidos

A cadeia de um manipulador pode ser modelada como um sistema de corpos rígidos. A localização de cada corpo rígido é completamente descrita por sua posição e orientação.

A posição pode ser representada pelas coordenadas de um ponto fixo arbitrário em relação ao corpo rígido. Na Figura 7 - Posição e orientação de um corpo rígido (33)., $O - xyz$ representa um sistema de coordenadas fixo em relação ao chão e O' representa um ponto arbitrário fixo ao corpo rígido. Neste exemplo a posição do corpo rígido é representada em referência ao sistema de coordenadas fixo no ponto $O - xyz$ da forma:

$$\boldsymbol{x_0} = [x_0 \quad y_0 \quad z_0]^T \tag{1}$$

onde $\boldsymbol{x_0}$ é um vetor 3x1.

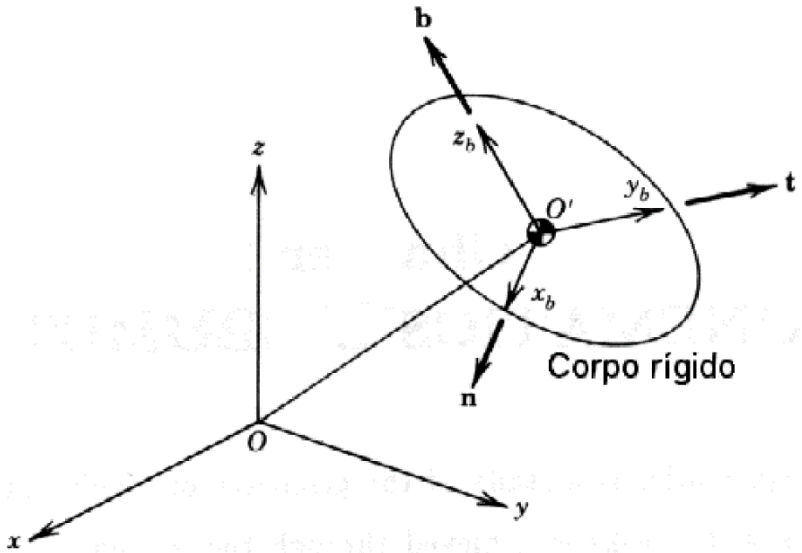

Figura 7 - Posição e orientação de um corpo rígido (33).

Para representar a orientação do corpo rígido, três eixos de coordenadas $\boldsymbol{x_b}$, $\boldsymbol{y_b}$ e $\boldsymbol{z_b}$ são adicionados, como mostrado na Figura 7 - Posição e orientação de um corpo rígido (33).. Estes eixos formam outro sistema de coordenadas denominado $O' - x_b y_b z_b$, que se move com o corpo rígido. A orientação do corpo agora pode ser representada pela direção dos três eixos de coordenadas. Sejam $\boldsymbol{n}$, $\boldsymbol{t}$ e $\boldsymbol{b}$ vetores unitários que apontam na direção dos eixos de coordenadas $\boldsymbol{x_b}$, $\boldsymbol{y_b}$ e $\boldsymbol{z_b}$, respectivamente. Os componentes de cada vetor unitário são cossenos direcionais de cada eixo de coordenada projetado no sistema de coordenadas fixo $O - xyz$. Por conveniência combinam-se os três vetores em uma matriz ortonormal $\boldsymbol{R}$ 3x3:

$$\boldsymbol{R} = [\boldsymbol{n} \quad \boldsymbol{t} \quad \boldsymbol{b}] \tag{2}$$

A matriz $\boldsymbol{R}$ descreve completamente a orientação do corpo rígido em referência ao sistema de coordenadas fixo $O - xyz$.

3.2.2. Transformação dos sistemas de coordenadas

A posição de um ponto arbitrário $\boldsymbol{P}$ pode ser representada tanto pelo sistema de coordenadas $O - xyz$ quanto pelo sistema $O' - x_b y_b z_b$.

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}$$

$$\boldsymbol{x^b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4}$$

Nas Equações (3) e (4), $\boldsymbol{x}$ é a representação no sistema fixo ao chão e $\boldsymbol{x^b}$ é a representação no sistema fixo ao corpo rígido.

Como mostrado na Figura 8, o ponto $\boldsymbol{P}$ pode ser obtido através de pontos de $\boldsymbol{O}$, $\boldsymbol{A}$ e $\boldsymbol{B}$. Isto é representada matematicamente pela relação:

$$\boldsymbol{OP} = \boldsymbol{OO'} + \boldsymbol{O'A} + \boldsymbol{AB} + \boldsymbol{BP} \tag{5}$$

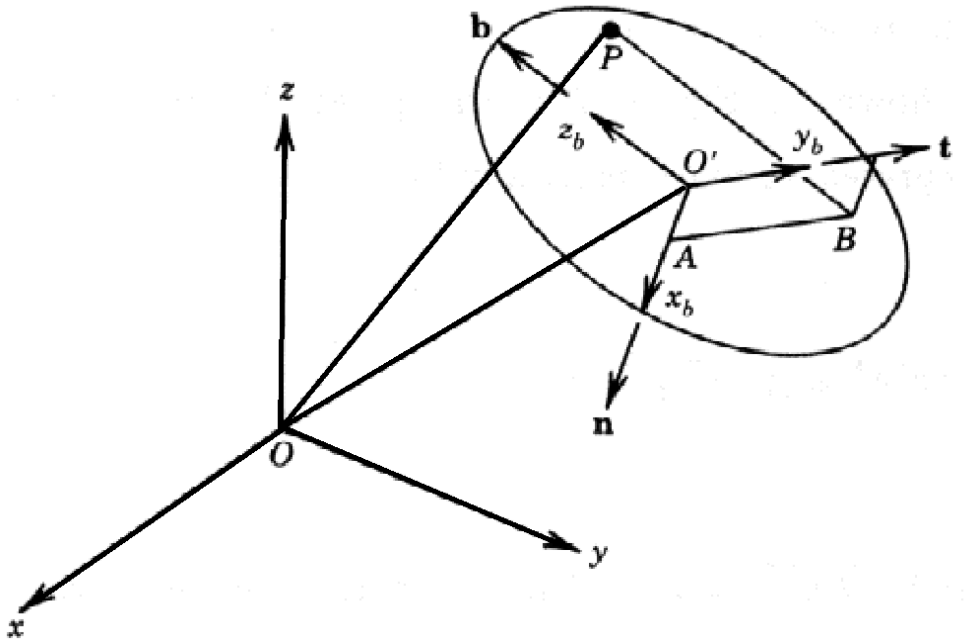

Figura 8 - Transformação do sistema de coordenadas (33).

onde $\boldsymbol{OP} = \boldsymbol{x}$ e $\boldsymbol{OO'} = \boldsymbol{x_0}$. Note-se que os vetores $\boldsymbol{O'A}$, $\boldsymbol{AB}$ e $\boldsymbol{BP}$ são paralelos aos vetores unitários $\boldsymbol{n}$, $\boldsymbol{t}$ e $\boldsymbol{b}$, respectivamente, e que seus comprimentos são dados por u, v e w. Então, é possível reescrever a expressão anterior como:

$$\boldsymbol{x} = \boldsymbol{x_0} + u\boldsymbol{n} + v\boldsymbol{t} + w\boldsymbol{b} \tag{6}$$

Utilizando as Equações (2) e (4), tem-se:

$$\boldsymbol{x} = \boldsymbol{x_0} + \boldsymbol{R}\boldsymbol{x^b} \tag{7}$$

A Equação (7) fornece a transformação de sistema de coordenadas desejado (da coordenada do corpo rígido $\boldsymbol{x^b}$ para coordenada fixa $\boldsymbol{x}$).

3.2.3. Transformações homogêneas

Nesta seção será apresentado um método para representar transformações de sistemas de coordenadas de uma forma mais compacta (34).

Observando a Equação (7) é possível afirmar que o primeiro termo do lado direito representa a translação da transformação, ou seja, o deslocamento espacial, enquanto o segundo termo representa a parcela de rotação. De forma a obter uma forma simplificada de representação, na qual os termos de translação e rotação

possam ser representados em um único termo matricial, definem-se os vetores 4x1:

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \boldsymbol{X^b} = \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{8}$$

e a matriz 4x4:

$$\boldsymbol{A} = \begin{bmatrix} & \boldsymbol{R} & & \boldsymbol{x_0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

A Equação (7) pode então ser escrita como:

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{X^b} \tag{10}$$

ou seja,

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} & \boldsymbol{R} & & \boldsymbol{x_0} \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{11}$$

Observe-se que a matriz 4x4 $\boldsymbol{A}$ representa a posição e a orientação do sistema de coordenadas $O - x_b y_b z_b$. Os dois termos no lado direito da Equação (7) são reduzidos a um termo único na Equação (10). A transformação de coordenadas dada pela Equação (10) é denominada transformação homogênea.

A compactação da transformação homogênea é particularmente vantajosa quando há transformações consecutivas. Seja $O'' - x_c y_c z_c$ um outro sistema de coordenadas, como mostrado na Figura 9.

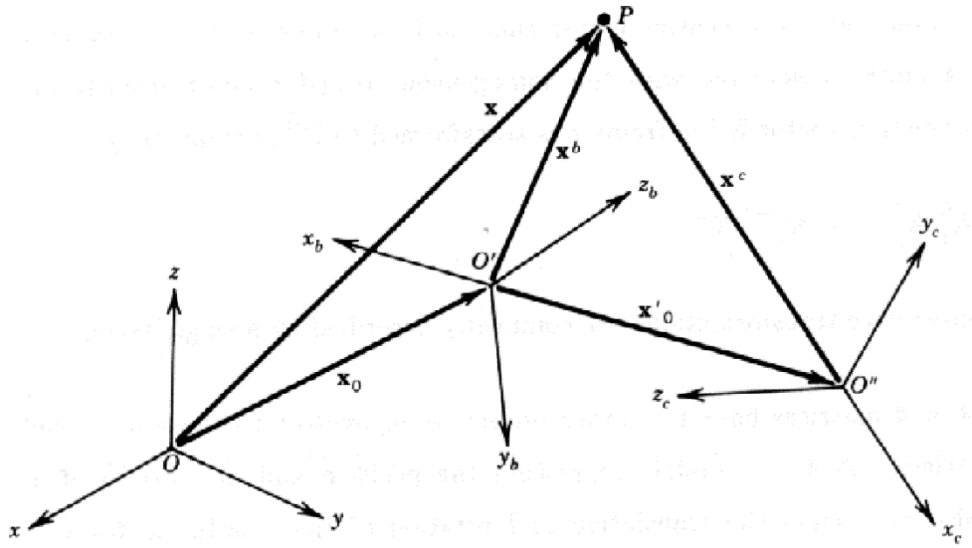

Figura 9 - Consecutivas transformações de sistemas de coordenadas (33).

Então

$$\boldsymbol{x^b = x'_0 + R' x^c} \tag{12}$$

onde $\boldsymbol{x'_0}$ e $\boldsymbol{R'}$ são um vetor 3x1 e uma matriz 3x3 associados com a transformação de $\boldsymbol{x^c}$ pra $\boldsymbol{x^b}$. Substituindo-se a Equação (12) na Equação (7):

$$\boldsymbol{x = x_0 + R x'_0 + R R' x^c} \tag{13}$$

Existem agora três termos do lado direito da Equação (13). Como a transformação é repetida, o número de termos no lado direito aumenta. Em geral, n transformações de coordenadas consecutivas conduzem a um polinómio de enésima ordem, composto de $(n+1)$ termos não-homogêneos. A transformação homogênea que utiliza a matriz 4x4, por outro lado, proporciona uma forma compacta, que representa qualquer transformação consecutiva com um termo único. Considerem-se n transformações consecutivas do sistema de coordenadas n de volta ao sistema 0. Seja $\boldsymbol{A_i^{i-1}}$- a matriz 4x4 associada com a transformação homogênea do sistema i para o sistema $i-1$. Um vetor de posição $\boldsymbol{X^n}$ do sistema de coordenadas n é transformado para $\boldsymbol{X^0}$ no sistema 0 pela equação:

$$\boldsymbol{X^0 = A_1^0 A_2^1 A_3^2 \dots A_n^{n-1} X^n} \tag{14}$$

Com isso as transformações consecutivas podem ser compactamente descritas por um único termo.

3.2.4.
Cadeia cinemática aberta

A transformação homogênea é utilizada para descrever a posição e a orientação de cada *link* de um manipulador.

Um manipulador é basicamente uma série de corpos rígidos em uma estrutura cinemática. A Figura 10 mostra um braço manipulador modelado como uma cadeia de corpos rígidos em série. Tal cadeia com uma estrutura serial ou *open loop* é referida como uma cadeia cinemática aberta, característica da maioria dos robôs industriais existentes.

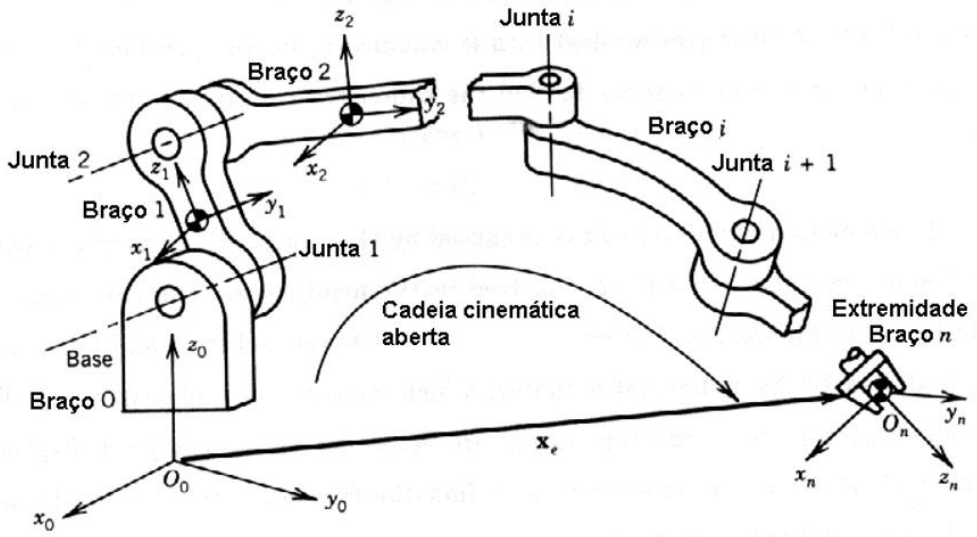

Figura 10 - Cadeia cinemática aberta (34).

Cada *link* (elo) de uma cadeia cinemática aberta pode ser numerado de 0 (base) a n (*link* mais distante), como mostrado na Figura 10. Uma vez que um manipulador executa uma tarefa por meio do movimento de um *end effector* (efetuador) ligado ao último elo, o interesse primário é analisar o movimento deste. Para representar a posição e orientação do efetuador, atribui-se um sistema de coordenadas $O_n - x_n y_n z_n$ ao último elo. A localização do sistema de coordenadas é descrita com referência a um outro sistema $O_0 - x_0 y_0 z_0$, fixado ao elo da base, como mostrado na Figura 10.

O movimento do efetuador é causado por movimentos dos elos intermediários entre o a base e o último elo. Assim, a localização final do efetuador pode ser determinada por exame da posição e orientação de cada elo em série desde a base até a ponta. Define-se, assim, um sistema de coordenadas $O_i - x_i y_i z_i$ para cada elo i. A posição e orientação do sistema $O_i - x_i y_i z_i$ em

relação ao anterior $O_{i-1} - x_{i-1}y_{i-1}z_{i-1}$ são descritas por meio da transformação homogênea entre os dois sistemas de coordenadas. A posição e a orientação do efetuador são então obtidas pelas transformações homogêneas consecutivas do último sistema de coordenadas até a base. Sendo o manipulador uma cadeia cinemática aberta, aplicam-se as transformações em série para encontrar a localização final do efetuador em relação à coordenada da base.

O movimento relativo de elos adjacentes é causado pelo movimento da junta que conecta os dois elos. Há um total de i juntas em um manipulador com $(i + 1)$ elos, tal como ilustrado na Figura 10. Por convenção, é comum referir-se à junta entre o elo $i - 1$ e o elo i como junta i. Cada junta é movida por um atuador individual, como um motor por exemplo. Assim, a posição e a orientação final de efetuador são determinadas pelos deslocamentos de i juntas.

A próxima seção seguinte aborda a relação funcional entre a localização do efetuador e os deslocamentos comuns de forma sistemática.

3.2.5. Notação de Denavit-Hartenberg

Nesta seção, será discutida a descrição da relação cinemática entre um par de *links* adjacentes envolvidos em uma cadeia cinemática aberta. A notação *Denavit-Hartenberg* (DH) é introduzida como uma forma sistemática de descrever essa relação (35). O método baseia-se na representação da matriz de posição e orientação do corpo rígido e faz uso de um número mínimo de parâmetros para descrever completamente a relação cinemática.

A Figura 11 mostra um par de *links* adjacentes $i - 1$ e i, e suas juntas associadas $i - 1$, i e $i + 1$. A linha H_iO_i é a normal comum aos eixos das juntas i e $i + 1$. A relação entre os dois *links* é descrita pela posição e orientação relativas dos sistemas de coordenadas dos dois *links*. Na notação de *Denavit-Hartenberg*, a origem do i-ésimo sistema de coordenadas O_i está localizado na intersecção dos eixos da junta $i + 1$ com a normal comum entre os eixos das juntas i e $i + 1$, como mostrado na Figura 11. Observe-se que a coordenada do *link* i situa-se na junta $i + 1$, ao invés da junta i. O eixo $\boldsymbol{x_i}$ é direcionado ao longo da linha de extensão da normal comum, enquanto que o eixo $\boldsymbol{z_i}$ situa-se ao longo do eixo da junta $i + 1$. Finalmente, o eixo $\boldsymbol{y_i}$ é escolhido de tal modo que o sistema de

coordenadas $O_i - x_i y_i z_i$ resultante forme um sistema de coordenadas obedecendo a regra da mão direita (36).

A localização relativa de dois sistemas de coordenadas pode ser totalmente determinada por quatro parâmetros:

- a_i – comprimento da normal comum;
- d_i – distância entre a origem O_{i-1} e o ponto H_i;
- α_i – ângulo entre o eixo de atuação da junta i, ou seja, z_{i-1} e o eixo z_i, seguindo a regra da mão direita;
- θ_i – o ângulo entre o eixo x_{i-1} e a normal comum $H_i O_i$, ou seja, x_i, também obedecendo a regra da mão direita.

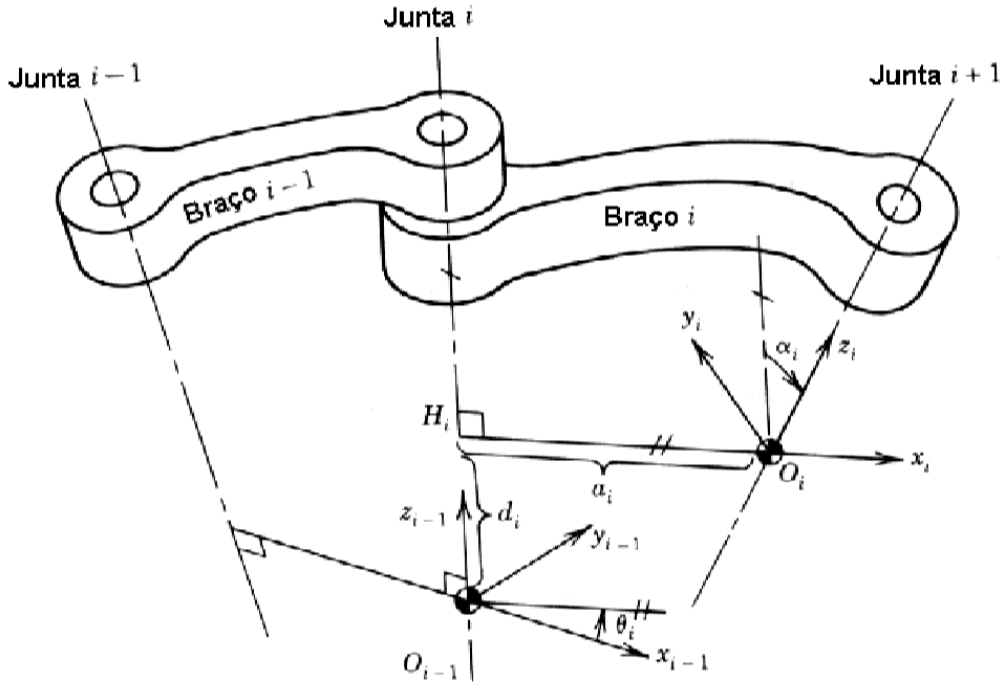

Figura 11 - A notação de Denavit-Hartenberg (34).

Os parâmetros a_i e α_i são constantes e são determinados pela geometria do elo: a_i representa o comprimento do link e α_i é o ângulo de torção entre os dois eixos comuns. Um dos outros dois parâmetros d_i e θ_i varia conforme o movimento das juntas.

Existem dois tipos de mecanismos de junta em braços mecânicos: juntas de revolução, em que os *links* adjacentes rodam um em relação ao outro em torno de um eixo comum, e juntas prismáticas, em que os links adjacentes deslizam linearmente um com o outro ao longo do eixo da junta. Para uma junta de

revolução, o parâmetro θ_i é a variável que representa o deslocamento da junta, enquanto o parâmetro d_i é constante. Para uma junta prismática, por outro lado, o parâmetro d_i é a variável que representa o deslocamento da junta, enquanto θ_i é constante.

Fazendo uso do conceito de transformação homogênea, a transformação do sistema de coordenadas O_{i-1} para o sistema O_i pode ser representado por duas translações (a_i e d_i) e duas rotações (θ_i e α_i).

$$\boldsymbol{A_i^{i-1}} = Rot_{z_i\theta_i} Trans_{z_i d_i} Trans_{x_i a_i} Rot_{x_i \alpha_i}$$

$$\boldsymbol{A_i^{i-1}} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A_i^{i-1}} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

A matriz $\boldsymbol{A_i^{i-1}}$representa a posição e orientação do sistema de coordenadas i relativamente ao sistema $i-1$. Os primeiros três vetores coluna contém os cossenos diretores dos eixos de coordenadas do sistema i, enquanto que o último vetor coluna representa a posição da origem O_i.

A notação de *Denavit-Hartenberg* possibilita obter de forma padronizada e simplificada a posição e orientação de qualquer ponto do manipulador em relação a uma coordenada de referência, já que, como visto em seções anteriores, um manipulador de n elos pode ser representado por uma sequência de multiplicações matriciais que representam as transformações consecutivas de cada sistema de coordenadas, segundo a equação:

$$\boldsymbol{T} = \boldsymbol{A_1^0 A_2^1 A_3^2 \dots A_n^{n-1}} \quad (16)$$

onde $\boldsymbol{T}$ é a matriz 4x4 que representa a posição e orientação do último *link* em relação à coordenada de referência.

3.2.6. Escolha de um sistema de coordenadas de referência.

Antes de se escolher um ponto especifico, fez-se necessário definir o tipo de base que seria utilizada. Existem dois tipos:

- Base fixa: referência presa ao chão, de forma que perturbações externas não sejam facilmente capazes de deslocar a coordenada de referência da posição determinada. Este tipo é o mais usado em robôs industriais e braços robóticos em geral, mas em robôs móveis a aplicação é mínima. Uma das aplicações possíveisseriam os pés do robô humanóide, mas a

escolha destes como referência implica na necessidade de modelar diversos estados em que os pés não se comportam como base fixa.

- Base flutuante: referência não fixa, geralmente associada a uma central inercial, capaz de estimar os deslocamentos sofridos pela base e compensá-los atraves da abstração de *links* virtuais que simulam os movimentos observados pelos sensores (37) (Figura 12). No robô Atlas existe uma central inercial próxima à pelvis que também é um otimo ponto para dividir o robô em manipuladores.

Com este tipo de base, é possível utilizar sempre um mesmo modelo, pois os sensores de força e torque presentes nas extremidades do robô serão responsáveis por provocar atuações com o objetivo de manter o robô em estado de equilibrio, como mostrado na Figura 13 (38).

A central inercial permite que, por meio das medidas dos acelerômetros e giroscópios, sejam estimadas variações de posição e de orientação da pélvis em relação a um referencial inercial, ou seja, neste caso similar a uma base fixa. Conforme mostrado na Figura 12, a equivalência se faz por meio da adição de seis juntas virtuais, três rotativas e três prismáticas. Tal escolha torna possível replicar qualquer movimento virtual da pélvis pois envolve todos os possíveis deslocamentos e rotações nos três eixos de coordenadas (37).

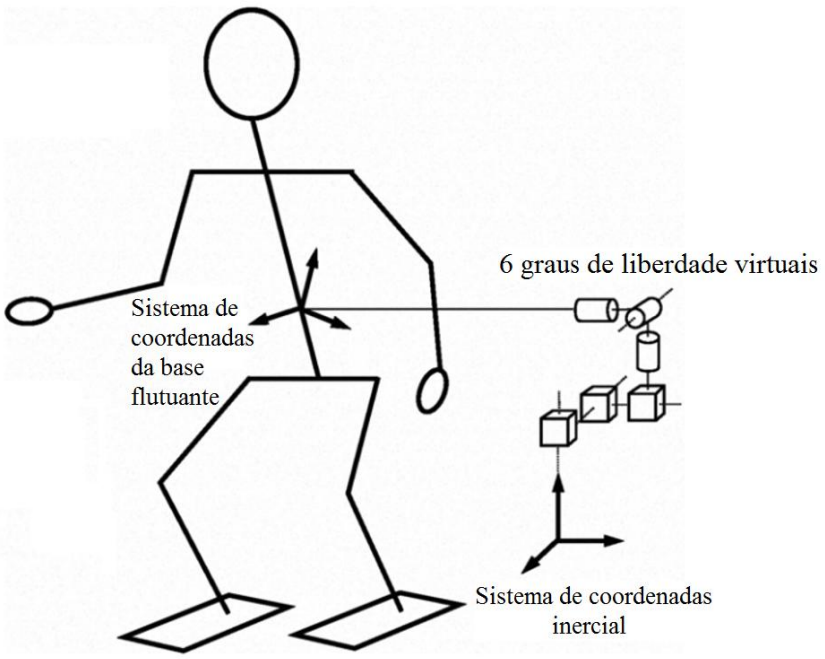

Figura 12 - Sistema de referência virtual possibilitado pela utilização de uma central inercial (37).

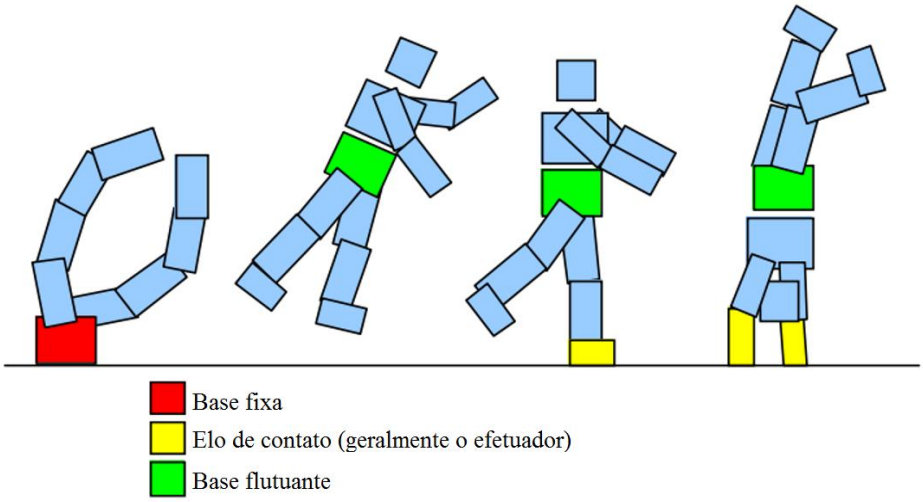

Figura 13 - Exemplo de mecanismos de base fixa e de base flutuante em diversas configurações (38).

A Figura 14 mostra o local selecionado como referência e o robô dividido em cinco manipuladores. Na parte inferior tem-se *R_leg* e *L_leg*, pernas direita e esquerda, respectivamente. Na parte superior tem-se o *Torso*, que faz parte de outros três manipuladores: *R_arm*, braço direito, *Head*, cabeça e *L_arm*, braço esquerdo. O *Torso* entra nas equações da cinemática de cada um dos membros superiores inicialmente de forma independente, ou seja, como um sexto manipulador, mas, quando a dinâmica dos manipuladores for modelada, cada manipulador contribuirá para o cálculo do torque a ser aplicado. Assim, é mais conveniente representar o Torso como parte dos três manipuladores.

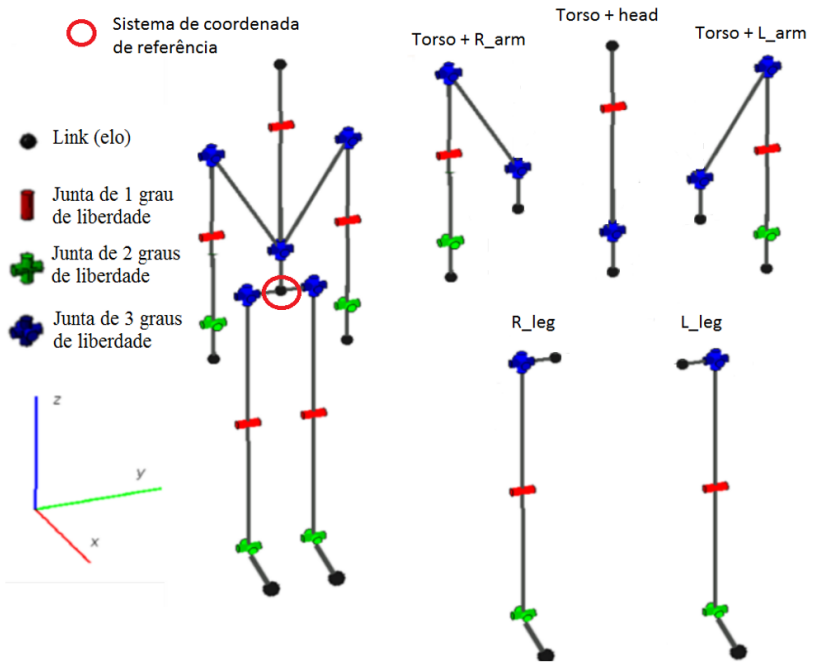

Figura 14 - Referência escolhida e manipuladores gerados a partir da escolha.

Com os manipuladores determinados, é possível determinar a posição e orientação de qualquer parte do manipulador para um conjunto de posições das juntas. Entretanto, o cálculo inverso seria muito mais útil, ou seja, determinar o conjunto de posições das juntas capaz de proporcionar uma determinada posição e orientação

3.3. Cinemática inversa

A cinemática inversa permite calcular os valores das coordenadas articulares com base na posição e orientação da ponta do objeto. O processo de resolução de um problema tradicional de cinemática inversa parte das coordenadas dos pontos inicial e final, dos comprimentos de todos os segmentos envolvidos e dos tipos de juntas. O conhecimento destas variáveis é suficiente para a descoberta de pelo menos uma solução.

Um manipulador deve ter pelo menos seis graus de liberdade para localizar o seu efetuador em um ponto arbitrário com uma orientação arbitrária no espaço. Se tiver mais do que seis graus de liberdade – caso de um braço humano, por exemplo –, pode existir um número muito grande de soluções. No caso do robô humanoide objeto deste trabalho, o número de graus de liberdade é superior a seis, o que torna interessante investigar outra metodologia (vide Cap. 4) que possibilite realizar de forma completamente autônoma o planejamento da trajetória.

A próxima seção tratará da velocidade com que o efetuador se move, pois, para movê-lo em uma direção especifica com uma velocidade controlada é necessário coordenar os movimentos de juntas individuais.

3.4. Matriz jacobiana

As velocidades observadas no efetuador são a velocidade linear, que corresponde a um deslocamento da posição, e a velocidade angular, referente à translação da orientação (39).

Para controlar a velocidade linear, o interesse primário é observar os pequenos movimentos do efetuador perto da posição atual. O movimento linear infinitesimal pode ser determinado pela derivada da equação da cinemática direta. Para um manipulador de n juntas, tem-se:

$$\begin{aligned} dx &= \frac{\partial x}{\partial q_1} dq_1 + \frac{\partial x}{\partial q_2} dq_2 \ldots \frac{\partial x}{\partial q_n} dq_n \\ dy &= \frac{\partial y}{\partial q_1} dq_1 + \frac{\partial y}{\partial q_2} dq_2 \ldots \frac{\partial y}{\partial q_n} dq_n \\ dz &= \frac{\partial z}{\partial q_1} dq_1 + \frac{\partial z}{\partial q_2} dq_2 \ldots \frac{\partial z}{\partial q_n} dq_n \end{aligned} \tag{17}$$

em forma vetorial:

$$\boldsymbol{v} = \dot{\boldsymbol{x}} = \mathbf{J}_{\boldsymbol{L}} \dot{\boldsymbol{q}} \tag{18}$$

onde $\dot{\boldsymbol{x}}$ e $\dot{\boldsymbol{q}}$ são, respectivamente, variações infinitesimais da posição do efetuador e dos ângulos ou deslocamentos das juntas, de forma que, como observado na seção sobre DH, se a i-ésima junta for prismática, e portanto variar um deslocamento, $q_i = d_i$, e se a i-ésima junta for rotativa, e portanto variar um ângulo, $q_i = \theta_i$. Os vetores $\dot{\boldsymbol{x}}$ e $\dot{\boldsymbol{q}}$ deste mesmo manipulador de n juntas são do tipo:

$$\dot{\boldsymbol{x}} = [dx\ dy\ dz]^T \qquad \dot{\boldsymbol{q}} = [dq_1\ dq_2 \ldots\ dq_n]^T \tag{19}$$

A matriz 3xn $\mathbf{J}_{\boldsymbol{L}}$ é a matriz jacobiana linear que compreende todas as derivadas parciais da posição do efetuador para cada deslocamento das n junta do manipulador genérico:

$$\mathbf{J}_{\boldsymbol{L}} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix} \tag{20}$$

Para computar a velocidade angular, avalia-se a contribuição de cada tipo de junta à velocidade angular. Como juntas prismáticas não são capazes de gerar esse tipo de velocidade, sua contribuição pode ser desconsiderada. Já as juntas rotativas têm fundamental importância na geração de velocidade angular. A

Equação (21), abaixo, forma a jacobiana angular de um manipulador genérico com n juntas.

$$\mathbf{J}_{\omega} = [\rho_1 \boldsymbol{z_0} \quad \rho_2 \boldsymbol{z_1} \quad \cdots \quad \rho_n \boldsymbol{z_{n-1}}] \tag{21}$$

A variável ρ pode receber dois valores: 0 (zero), se a junta associada a esse número for prismática, e 1 (um), se a junta for de revolução. $\boldsymbol{z_i}$ corresponde ao eixo z do i-ésimo sistema de coordenadas referenciados a base, ou seja, os primeiros três elementos da terceira coluna da matriz observada na Equação (16) que representa a posição e orientação de uma junta genérica em relação a base.

A matriz 3xn $\mathbf{J_{\omega}}$ representa a velocidade angular do efetuador de um manipulador genérico de n juntas.

Juntando a velocidade linear com a angular tem-se:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \mathrm{J}_L \\ \mathrm{J}_{\omega} \end{bmatrix} \dot{q} = \mathrm{J}\dot{q} \tag{22}$$

A Equação (22) refere-se a manipulador ideal, ou seja, não leva em consideração efeitos dinâmicos nem gravidade.

3.5. Modelagem da dinâmica do robô

O comportamento dinâmico é descrito em termos da taxa de variação da configuração do braço em relação aos torques exercidos nas juntas pelos atuadores (40). Esta relação pode ser expressa por um conjunto de equações diferenciais, chamadas equações de movimento, que governam a resposta dinâmica do braço para valores de torques nas juntas. Dois métodos podem ser utilizados para obter as equações de movimento: *Newton-Euler* e *Euler-Lagrange* (41). A formulação *Newton-Euler* é oriunda da interpretação direta da Segunda Lei de Newton do Movimento, que descreve sistemas dinâmicos em termos de força e momento. As equações incorporam todas as forças e momentos atuando nos elos individuais do braço, incluindo as forças de acoplamento e momentos entre os elos. As equações incluem as forças de restrição que atuam entre elos adjacentes. Assim, operações aritméticas adicionais são necessárias para eliminar esses termos e obter relações explícitas entre os torques das juntas e do movimento resultante do deslocamento das juntas.

Na formulação de *Euler-Lagrange* (42), o comportamento dinâmico do sistema é descrito em termos de trabalho e de energia por meio de coordenadas generalizadas. Todas as forças sem trabalho e as forças de restrição são automaticamente eliminados. As equações resultantes são geralmente compactas e fornecem uma expressão de forma fechada em termos de torques e deslocamentos das juntas. Além disso, a derivação é mais simples e sistemática do que no método *Newton-Euler*.

3.5.1. Método de *Euler-Lagrange*

Sejam $q_1, \dots, q_n$ as coordenadas generalizadas que localizam completamente um sistema dinâmico. Sejam $\boldsymbol{T}$ e $\boldsymbol{U}$ a energia cinética e a energia potencial armazenada no sistema dinâmico. Define-se o Lagrangiano $\boldsymbol{L}$ por:

$$\boldsymbol{L}(q_i, \dot{q}_i) = \boldsymbol{T} - \boldsymbol{U} \tag{23}$$

Note-se que, já que as energias cinética e potencial são função de q_i e $\dot{q}_i$ $(i = 1, \dots, n)$, o Lagrangiano $\boldsymbol{L}$ também o é. Usando o Lagrangiano, as equações do movimento do sistema dinâmico são dadas por:

$$\frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{q}_i} - \frac{\partial \boldsymbol{L}}{\partial q_i} = \boldsymbol{Q_i} \tag{24}$$

onde $\boldsymbol{Q_i}$ é a força generalizada correspondente à coordenada generalizada q_i. As forças generalizadas podem ser identificadas considerando-se o trabalho virtual realizado por forças não consecutivas agindo sobre o sistema.

A partir das Equações (23) e (24) é possível derivar as equações do manipulador. Inicialmente, será derivado o termo relativo a energia cinética armazenada em um elo individual do braço. Como mostrado na Figura 15, para o i-ésimo elo, $\boldsymbol{v_{ci}}$ representa um vetor 3x1 de velocidade linear do centro de massa, e $\boldsymbol{\omega_i}$, o vetor 3x1 da velocidade angular, ambos com sistemas de coordenadas referenciados na base. Ambos os dados são obtidos pelo cálculo da matriz jacobiana do centro de massa referenciado também na base. A energia cinética do elo i é dada por

$$\boldsymbol{T_i} = \frac{1}{2} m_i \boldsymbol{v_{ci}}^T \boldsymbol{v_{ci}} + \frac{1}{2} \boldsymbol{\omega_i}^T \boldsymbol{I_i} \boldsymbol{\omega_i} \tag{25}$$

onde m_i é a massa do elo e $\boldsymbol{I_i}$ é a matriz de inércia no centroide expressa nas coordenadas da base.

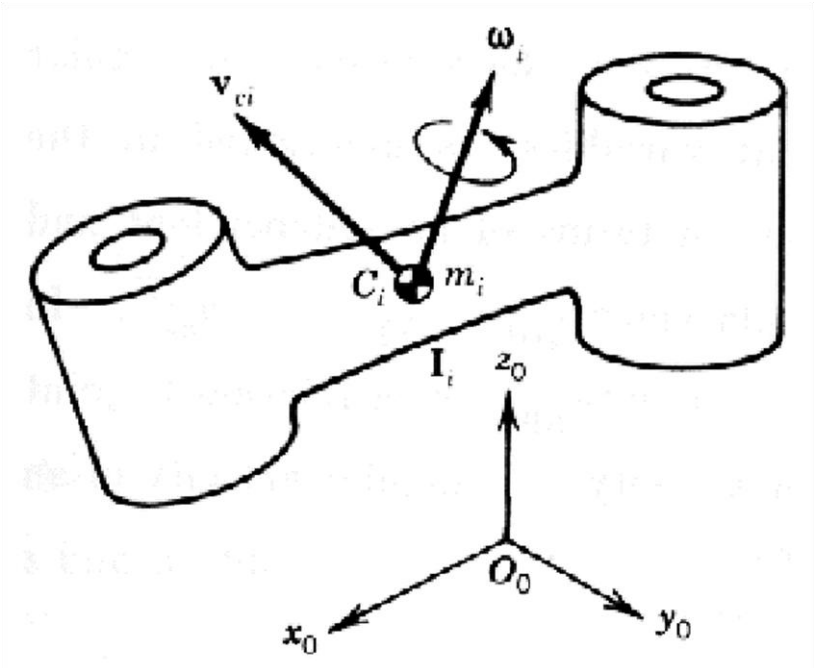

Figura 15 - Velocidade linear e angular do centroide do elo $\boldsymbol{i}$ (42).

O primeiro termo da Equação (25) computa a energia cinética resultante do movimento de translação da massa m_i; o segundo termo representa a energia cinética resultante do movimento de rotação ao redor do centroide. A energia cinética total armazenada em toda a cadeia que compõe o braço robótico é dada pela soma das energias cinéticas de cada centroide de cada elo que compõe o braço:

$$\boldsymbol{T} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{T_i} \tag{26}$$

Na seção 3.4, analisaram-se a velocidade linear e angular de um efetuador em relação à velocidade das juntas. Considerando, agora, os centros de massa de cada elo como se fossem os efetuadores, é possível empregar o mesmo método para calcular a velocidade linear e angular do centro de massa do elo. Expandindo a Equação (22) e substituindo o resultado nas Equações (25) e (26), obtém-se:

$$\boldsymbol{T} = \frac{1}{2}\sum_{i=1}^{n}\left(m_i \dot{q}^T \mathbf{J}_{\boldsymbol{L}}^{(\boldsymbol{i})\boldsymbol{T}} \mathbf{J}_{\boldsymbol{L}}^{(\boldsymbol{i})} \dot{q} + \dot{q}^T \mathbf{J}_{\boldsymbol{\omega}}^{(\mathbf{i})\boldsymbol{T}} \boldsymbol{I}_i \mathbf{J}_{\boldsymbol{\omega}}^{(\boldsymbol{i})} \dot{q}\right) \tag{27}$$

onde $\mathbf{J}_{\boldsymbol{Lj}}^{(\boldsymbol{i})}$ e $\mathbf{J}_{\boldsymbol{\omega j}}^{(\boldsymbol{i})}$ são a j-ésima coluna do vetor 3xn da matriz jacobiana linear e angular do elo i, respectivamente.

Esta equação pode ser reescrita como:

$$\boldsymbol{T} = \frac{1}{2}\dot{q}^T \boldsymbol{H}\dot{q} \tag{28}$$

onde $\boldsymbol{H}$ é uma matriz nxn dada pela expressão:

$$\boldsymbol{H} = \sum_{i=1}^{n}\left(m_i \mathbf{J}_{\boldsymbol{L}}^{(\boldsymbol{i})\boldsymbol{T}} \mathbf{J}_{\boldsymbol{L}}^{(\boldsymbol{i})} + \mathbf{J}_{\boldsymbol{\omega}}^{(\mathbf{i})\boldsymbol{T}} \boldsymbol{I}_{\boldsymbol{i}} \mathbf{J}_{\boldsymbol{\omega}}^{(\boldsymbol{i})}\right) \tag{29}$$

A matriz $\boldsymbol{H}$, denominada matriz de inercia do manipulador, incorpora todas as propriedades da massa de toda a cadeia do braço e tem características similares às matrizes de inércia individuais: simétrica e definida positiva. Contudo a matriz de inercia do manipulador é dependente da configuração e representa as propriedades relativa a massas instantâneas de toda a cadeia na configuração atual.

Considerando $\boldsymbol{H_{ij}}$ as componentes $[i, j]$ da matriz de inercia $\boldsymbol{H}$ do manipulador, é possível reescrever a Equação (28) de uma forma escalar

$$\boldsymbol{T} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{H_{ij}}\dot{q}_i\dot{q}_j \tag{30}$$

Seja $\boldsymbol{g}$ o vector 3x1 representando a aceleração da gravidade e $\boldsymbol{r_{0,ci}}$ o vetor de posição extraído da equação da cinemática direta para centroide do i-ésimo elo, ambos com referência ao sistema de coordenadas da base. Em seguida, a energia potencial armazenada em todos elos da cadeia do braço robótico é dada por:

$$\boldsymbol{U} = \sum_{i=1}^{n} m_i \boldsymbol{g}^T \boldsymbol{r_{0,ci}} \tag{31}$$

Como o vetor posição do centroide $\boldsymbol{C_i}$ depende da configuração do braço, a equação da energia potencial é função de $q_1, \ldots, q_n$.

Forças generalizadas são responsáveis por todas as forças e momentos atuando sobre a articulação do braço, exceto as forças da gravidade e da inércia. Considere a situação em que atuadores exercem torques $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_n]^T$ nas juntas individuais e uma força e momento externos $\boldsymbol{F_{ext}}$ é aplicada na

extremidade do braço enquanto em contato com o ambiente. Forças generalizadas podem ser obtidas pelo cálculo do trabalho virtual $\boldsymbol{\delta Work}$ por elas realizado. Este é dado pelo deslocamento virtual das juntas, denominado δq, causado por um torque aplicado, e pelo deslocamento virtual do efetuador, denominado δp, causado por uma força externa. Como o deslocamento do efetuador é função do deslocamento das juntas, tal que $d\boldsymbol{p} = \mathbf{J}d\boldsymbol{q}$, é possível chegar a:

$$\boldsymbol{\delta Work} = \boldsymbol{\tau}^T \boldsymbol{\delta q} + \boldsymbol{F}_{\boldsymbol{ext}}^T \boldsymbol{\delta p} = \left(\boldsymbol{\tau} + \mathbf{J}^T \boldsymbol{F}_{\boldsymbol{ext}}\right)^T \boldsymbol{\delta q} \tag{32}$$

isto pode ser reescrito como:

$$\boldsymbol{\delta Work} = \boldsymbol{Q}^T \boldsymbol{\delta q} \tag{33}$$

onde $\boldsymbol{Q} = \boldsymbol{\tau} + \mathbf{J}^T \boldsymbol{F}_{\boldsymbol{ext}}$ representa as forças generalizadas [vide Equação (24)].

A partir das Equações (30) e (31), deriva-se a função de *Euler-Lagrange* e se obtêm as equações de movimento. O primeiro termo da Equação (24) resume-se ao cálculo para energia cinética, pois a potencial apresenta derivada zero. O termo é:

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{T}}{\partial \dot{q}_i}\right) = \frac{d}{dt}\left(\sum_{j=1}^{n} \boldsymbol{H}_{\boldsymbol{ij}} \dot{q}_j\right) = \sum_{j=1}^{n} \boldsymbol{H}_{\boldsymbol{ij}} \ddot{q}_j + \sum_{j=1}^{n} \frac{d\boldsymbol{H}_{\boldsymbol{ij}}}{dt} \dot{q}_j \tag{34}$$

Note-se que $\boldsymbol{H}_{\boldsymbol{ij}}$ é função de $q_1, \ldots, q_n$, de forma que a derivada de $\boldsymbol{H}_{\boldsymbol{ij}}$ é dada por:

$$\frac{d\boldsymbol{H}_{\boldsymbol{ij}}}{dt} = \sum_{k=1}^{n} \frac{\partial \boldsymbol{H}_{\boldsymbol{ij}}}{\partial q_k} \frac{dq_k}{dt} = \sum_{k=1}^{n} \frac{\partial \boldsymbol{H}_{\boldsymbol{ij}}}{\partial q_k} \dot{q}_k \tag{35}$$

O segundo termo da Equação (24) inclui a derivada parcial da energia cinética e da energia potencial gravitacional. A parcela relativa à energia cinética é dada por:

$$\frac{\partial \boldsymbol{T}}{\partial q_i} = \frac{\partial}{\partial q_i}\left(\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{H}_{\boldsymbol{jk}} \dot{q}_j \dot{q}_k\right) = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial \boldsymbol{H}_{\boldsymbol{jk}}}{\partial q_i} \dot{q}_j \dot{q}_k \tag{36}$$

Já que $\boldsymbol{H}_{\boldsymbol{ij}}$ depende de q_i, o cálculo da derivada parcial da energia potencial gravitacional da origem ao termo gravitacional $\boldsymbol{G}_{\boldsymbol{i}}$ é dado por:

$$\boldsymbol{G_i} = \frac{\partial \boldsymbol{U}}{\partial q_i} = \sum_{j=1}^{n} m_j \boldsymbol{g}^T \frac{\partial \boldsymbol{r_{0,cj}}}{\partial q_i} = \sum_{j=1}^{n} m_j \boldsymbol{g}^T \mathbf{J}_{\boldsymbol{Li}}^{(j)} \tag{37}$$

Como a derivada parcial do vetor de posição $\boldsymbol{r_{0,cj}}$ em relação a q_i é equivalente ao i-ésimo vetor coluna da matriz jacobiana $\mathbf{J}_{\boldsymbol{Li}}^{(j)}$ [Equação (22)], a substituição das Equações. (34), (36) e (37) na Equação (24) dá:

$$\sum_{j=1}^{n} H_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_j\dot{q}_k + G_i = Q_i \qquad i = 1, \dots, n \tag{38}$$

onde

$$\boldsymbol{H} = \sum_{i=1}^{n}\left(m_i \mathbf{J}_{\boldsymbol{L}}^{(i)\boldsymbol{T}} \mathbf{J}_{\boldsymbol{L}}^{(i)} + \mathbf{J}_{\boldsymbol{\omega}}^{(\mathrm{i})\boldsymbol{T}} \boldsymbol{I_i} \mathbf{J}_{\boldsymbol{\omega}}^{(i)}\right) \tag{39}$$

$$h_{ijk} = \frac{\partial \boldsymbol{H_{ij}}}{\partial q_k} - \frac{1}{2}\frac{\partial \boldsymbol{H}_{jk}}{\partial q_i} \tag{40}$$

e

$$\boldsymbol{G_i} = \sum_{j=1}^{n} m_j \boldsymbol{g}^T \mathbf{J}_{\boldsymbol{Li}}^{(j)} \tag{41}$$

Estas equações geram três termos que representam parcelas muito relevantes da dinâmica do manipulador. O primeiro termo é a matriz de inércia do manipulador $\boldsymbol{H}$, primordial para compensar torques extras necessários para retirar o braço da inércia ou desacelerá-lo para atingir um ponto desejado. Sem este termo não há como controlar a aceleração do manipulador.

A segunda parcela, que engloba vários termos, contém os denominados termos de Cristoffen $\boldsymbol{h_{ijk}}$, que em si representam as forças de coriolis e as forças centrífugas. Tais efeitos ocorrem simplesmente por um corpo A estar se movendo enquanto acoplado a um corpo B que também está se movendo. Com isto, há a necessidade de adicionar ou remover energia no corpo A para que este se mantenha estacionário em relação a B. Se B estivesse parado, tal efeito não seria observado.

A última parcela refere-se ao termo gravitacional, responsável por compensar as forças da gravidade, permitindo assim que o manipulador possa

permanecer em uma posição estacionaria bem como deslocar-se sem sucumbir à gravidade.

Vale lembrar, como indicado na Equação (33), que as equações da dinâmica levam em consideração forças externas que possam reagir com o manipulador. No modelo robótico Atlas, tais forças são adquiridas por quatro sensores de força e torque localizados nas extremidades dos braços e pernas do robô, mais precisamente *R_arm*, *L_arm*, *R_leg* e *L_leg*.

Com o modelo da dinâmica em mãos, pode-se definir agora uma malha de controle capaz de controlar o robô de forma precisa e estável.

3.6. Controle das juntas

O robô Atlas simulado possui controle de juntas do tipo PID. Para obter um controle rápido e com *overshoot* baixo é necessário ajustar os controladores de cada junta. Aplicações com baixas tolerâncias para *overshoot* fazem uso, geralmente, de controles do tipo PD, que são mais simples de sintonizar e apresentam resultados comparáveis ao PID, a menos do menor erro em regime permanente proporcionado pelo ganho integral.

Com este tipo de controle, é possível determinar as posições desejadas para as juntas, porém os efeitos dinâmicos provocam oscilações causadas pelos efeitos de coriolis e centrífugo bem como da parcela inercial.

O termo gravitacional é também uma dificuldade para que o controlador chegue à posição desejada, pois representa uma força não computada no momento da definição dos pesos do PD – provavelmente realizada em um ambiente sem efeitos dinâmicos. Vale lembra que qualquer força externa aplicada ao robô também prejudica o controle de forma similar ao termo gravitacional.

Os efeitos acima podem ser compensados pela utilização e um controle por torque computado.

3.7. Controle por torque computado

A solução de controle por torque computado tem por objetivo adicionar à malha de controle os efeitos dinâmicos de forma a compensá-los (42). A Figura

16 mostra a malha proposta, onde $\boldsymbol{q}$ representa a variável a ser controlada e $\boldsymbol{u}$ a saída do controle PD.

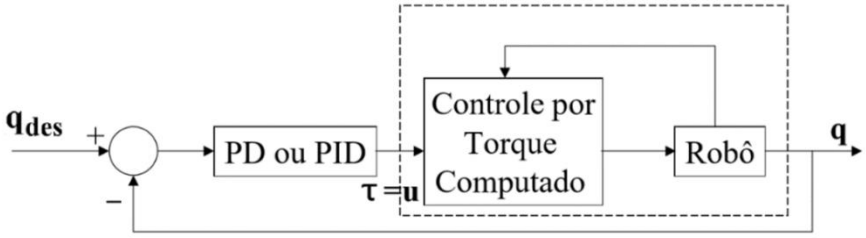

Figura 16 - Introdução do controle por torque computado na malha.

Para definir as forças a serem compensadas, será utilizada a Equação (38), que pode ser escrita como:

$$\tau = H(q)\ddot{q} + h(q,\dot{q}) + G(q) \tag{42}$$

como $\boldsymbol{H}$ é definida positiva, é possível utilizar a seguinte lei de controle:

$$\tau = H(q)u + h(q,\dot{q}) + G(q) \tag{43}$$

onde

$$u = \ddot{q} \tag{44}$$

Como todos os termos a serem compensados são função de $\boldsymbol{q}$ e $\dot{\boldsymbol{q}}$, que podem ser aferidos pelos sensores, o diagrama de blocos da malha realimentada final tem a seguinte forma (Figura 17):

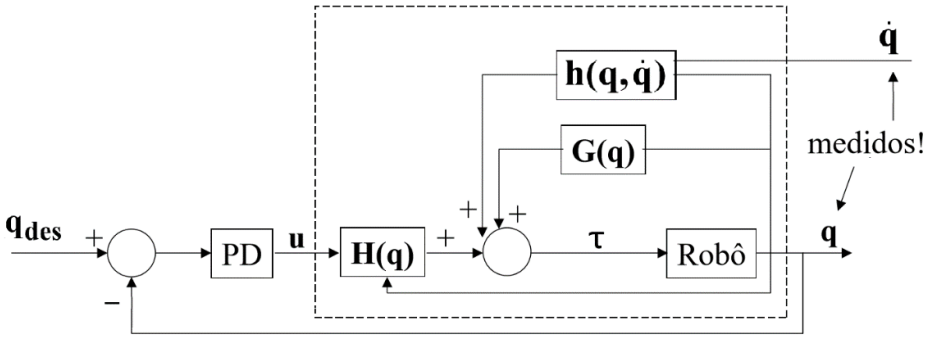

Figura 17 - Diagrama de blocos da malha do controle por torque computado.

A malha utilizada para o controle da posição das juntas contém as compensações necessárias (dos efeitos dinâmicos) para gerar uma determinada posição e velocidade (de cada junta).

É importante ressaltar que as forças externas geradas pelo contato dos pés ao chão são compensadas pela adição de torques iguais e contrários aos medidos

pelos sensores de força e torque nos pés, com isto, o robô é capaz de atingir uma postura bípede estável.

3.8. Análise da estabilidade bípede do robô humanoide.

A locomoção e a análise da estabilidade de um robô humanoide são elementos essenciais no desenvolvimento do robô. Esta análise é tipicamente definida de acordo com três planos (43):

- Sagital - define as caraterísticas de locomoção no eixo x do robô- por exemplo, andar para a frente e para trás;
- Frontal - define as caraterísticas de locomoção no eixo y do robô- por exemplo, andar para o lado;
- Transverso - define as caraterísticas de locomoção em torno do eixo z – por exemplo, rodar sobre si mesmo.

O plano sagital atravessa longitudinalmente o corpo, dividindo-o nas partes esquerda e direita. O plano frontal divide o corpo nas partes anterior e posterior. O plano transverso é o plano ortogonal aos dois planos anteriores. Este plano divide o corpo nas partes superior e inferior. A Figura 18 esquematiza os planos e lados referidos.

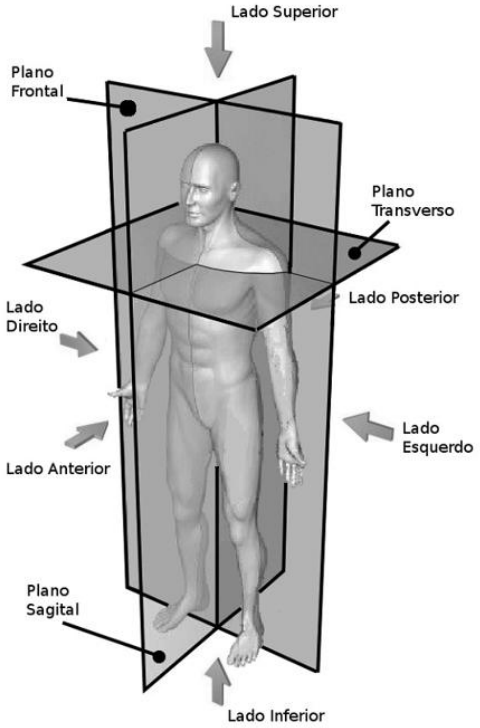

Figura 18 - Planos anatômicos de um robô humanoide (43).

3.8.1.
Configuração de locomoção de robôs

A locomoção bípede garante uma maior agilidade dos robôs para evitar obstáculos e permite uma maior facilidade na sua utilização em ambientes humanos. Quanto menor o número de pernas, mais complexo se torna o desenvolvimento da locomoção devido aos problemas que surgem com a estabilidade. Exemplos de configurações de robôs podem ser observados na Figura 19.

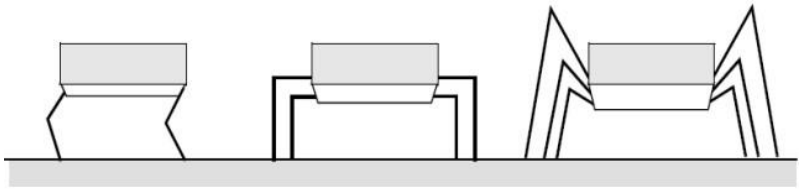

Figura 19 - Configuração bípede, quadrúpede e hexápode (43).

3.8.2.
Polígono de suporte

O polígono de suporte corresponde à superfície de contato entre o pé e o chão quando apenas um pé está no chão. Quando ambos os pés estão no chão, o

polígono de suporte é designado pelo polígono convexo que contém ambos os pés. Este polígono pode ser encontrado desenhando-se linhas retas que unem as partes exteriores de cada superfície de contato (Figura 20).

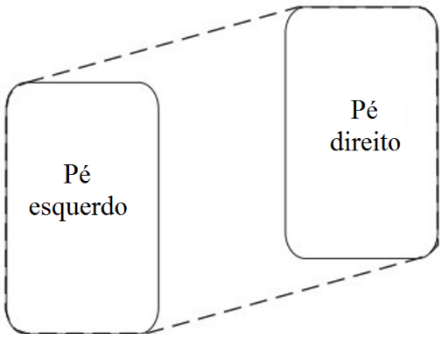

Figura 20 - Derivação do polígono de suporte no plano transverso.

3.8.3. Centro de Massa

O centro de massa do corpo é um critério de estabilidade largamente utilizado para o controle do equilíbrio estático. Um robô é considerado estaticamente estável quando a projeção no solo do seu centro de massa segundo a força da gravidade está contida no interior do seu polígono de suporte, como demonstrado na Figura 21 - Projeção do centro de massa e polígono de suporte (43).Figura 21.

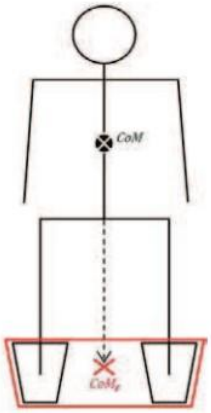

Figura 21 - Projeção do centro de massa e polígono de suporte (43).

3.8.4. Locomoção estática e dinâmica

A diferença entre a estabilidade das diferentes configurações apresentadas na Figura 19 pode ser explicada com os conceitos de equilíbrio estático e dinâmico. Como demonstrado na Figura 22, a locomoção estática considera que o robô está sempre numa posição estaticamente equilibrada: mesmo que pare em

qualquer momento, o robô permanecerá numa posição estável. Como a velocidade da locomoção estática é geralmente baixa, a projeção do centro de massa no plano horizontal é igual ao centro de pressão – ponto no plano de suporte do corpo onde atua a soma total das forças de contato. Para que o robô se mantenha estaticamente equilibrado, o centro de massa e o centro de pressão devem estar contidos no polígono de suporte. A locomoção dinâmica é usualmente mais rápida do que a estática. Nela, o centro de massa pode estar fora do polígono de suporte durante um tempo limitado, o que significa que o robô estaria estaticamente desequilibrado durante a fase de suporte com um pé, por exemplo.

Figura 22 - Equilíbrio estático e dinâmico (43).

3.8.5. Aplicação dos conceitos ao problema

O método de análise da posição do centro de massa é bastante útil para assegurar a estabilidade em locomoção estática. Atrelado ao controle, possibilita que o robô tenha liberdade de movimentos com grande probabilidade de êxito, ou seja, pouca chance de cair. Isso é possível pois uma combinação de juntas que seja proposta pode ser avaliada inicialmente pelas equações da cinemática do robô. Se o centro de massa se mantiver dentro do polígono de suporte, a configuração é aprovada.

A locomoção estática foi escolhida para o problema por apresentar menor risco de quedas. Além disso, uma locomoção dinâmica para um sistema robótico desta magnitude seria significativamente mais complexa e requereria o estudo de diversos pontos de estabilidade local para diversas combinações de juntas.

É importante ressaltar que a locomoção bípede é composta de um ciclo que alterna entre equilíbrio com dois pés, denominado duplo suporte, e com um pé, chamado de suporte simples.

3.9.
Ciclo de locomoção bípede

A Figura 23 mostra um exemplo de um ciclo de locomoção bípede, que consiste em duas fases de suporte duplo e duas fases de suporte simples uma em cada perna.

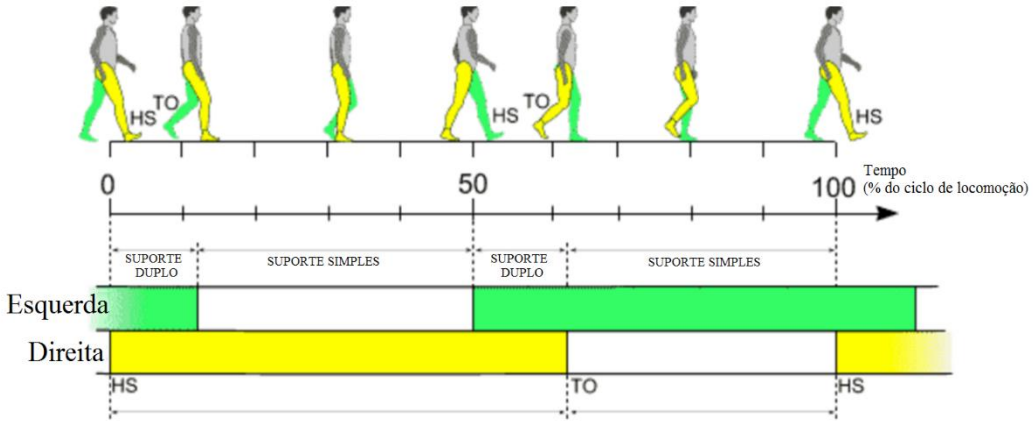

Figura 23 - Ciclo de locomoção bípede (43).

Inicia-se o movimento na fase de suporte duplo, em que há a necessidade de um movimento simultâneo entre as duas pernas para que nenhuma delas perca sustentação durante o movimento. O objetivo é mover o centro de massa do corpo para um dos pés, neste caso o direito, representado pela cor amarela. Vale lembrar que todo o movimento deve ser feito sem retirar o centro de massa de dentro do polígono de suporte; caso contrário, o robô tende a cair no chão.

Na etapa seguinte, já com o centro de massa dentro polígono correspondente à área que compõe o pé direito, pode-se retirar do chão o pé esquerdo e realizar um movimento para locomover o robô para a frente. Durante todo o percurso feito pela perna esquerda, o centro de massa não pode sair de dentro do polígono de suporte do pé direito; caso contrário, o robô tende a cair.

Com os dois pés no chão repete-se o primeiro movimento, mas desta vez deslocando o centro de massa para o outro pé, neste caso o pé esquerdo.

O penúltimo estagio é similar ao segundo, só que de forma inversa: o pé esquerdo está fixo ao chão e o direito se move.

Por fim, acaba-se por retornar ao primeiro estágio, configurando-se um ciclo de locomoção estática. É importante ressaltar que em todos os estágios, dada a

opção pela locomoção estática, o centro de massa não pode deixar o polígono de suporte respectivo de cada fase.

3.10. Determinação da trajetória

Em um manipulador robótico, por exemplo, são definidos alguns pontos onde o efetuador deve passar, determinando-se a seguir equações que liguem tais pontos. Com estas em mãos, é possível determinar uma série de pontos bem próximos pelos quais o atuador deve passar e, utilizando a cinemática inversa do manipulador, determinadas configurações das juntas para que cada ponto seja atingido e assim realizada uma trajetória. Entretanto, como já explicado, fica muito difícil utilizar a cinemática inversa para sistemas com mais de seis graus de liberdade. Além disso, no caso de um sistema dinâmico, seria necessário utilizar a dinâmica inversa para determinar não só as posições de cada junta, mas também suas velocidades e acelerações, tornando o cálculo bastante complexo. Por este motivo, é comum a utilização de métodos de inteligência computacional, como redes neurais (44, 45, 46), lógica fuzzy (47, 48, 49) e aprendizado por reforço (50, 51, 52), para determinar tais trajetórias e assim gerar um padrão de movimentos que possibilitem uma locomoção (estática, no caso). A Figura 24 demonstra como devem interagir o método de inteligência computacional e o controle do robô.

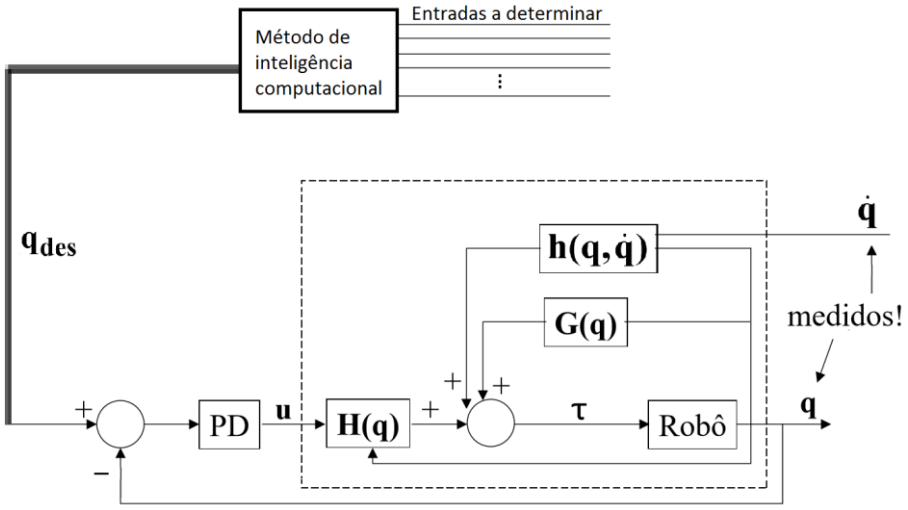

Figura 24 - Localização do método de inteligência computacional na malha de controle.

No capítulo seguinte será apresentado o método escolhido para o planejamento da trajetória de locomoção: aprendizado por reforço. Serão também

explicitados os conhecimentos necessários para a concepção do modelo e as adaptações necessárias para sua aplicação ao problema.

4.
Conceitos básicos para o desenvolvimento do método de planejamento da locomoção

No capítulo anterior foram desenvolvidos o modelo da dinâmica do robô e a malha de controle para que cada junta esteja em um ângulo definido. Entretanto, o ângulo a ser determinado depende da estipulação de uma trajetória para a cadeia cinemática a ser controlada.

Este capítulo apresenta um método baseado em aprendizado por reforço que permite encontrar uma determinada sequência de posições de ângulos para gerar um movimento de caminhada bípede.

A seção e subseções seguintes discorrem brevemente sobre métodos de aprendizado de máquina, de modo a justificar a opção pelo aprendizado por reforço.

4.1.
Aprendizado de máquina

Aprendizado de máquina é um campo que se originou do estudo do reconhecimento de padrões e da teoria do aprendizado computacional proveniente da inteligência artificial (53). O aprendizado de máquina explora o estudo e desenvolvimento de algoritmos que têm a capacidade de, entre outras funções, aprender ou realizar previsões a partir de uma base de dados (54).

Tais algoritmos são de implementação relativamente fácil e capazes de modelar problemas complexos sem que o usuário tenha conhecimento profundo do problema. Como ressaltado no capítulo anterior, o modelo da dinâmica inversa é consideravelmente complexo e por isso se justifica a utilização de métodos alternativos para solucionar o problema de definição da trajetória.

Os algoritmos de aprendizado de máquina podem ser divididos em três classes: supervisionado, não supervisionado e por reforço.

4.1.1.
Aprendizado supervisionado

No aprendizado supervisionado, as informações acerca das variáveis de entrada e saída contidas na base de dados são usadas para treinar o algoritmo – fase de aprendizado. Uma vez treinado, o algoritmo deve ser capaz de produzir uma saída adequada quando submetido a novos dados de entrada (55).

É importante ressaltar que o conjunto finito de dados de entrada e saída apresentado ao algoritmo deve ser uma amostra representativa de todas as possíveis combinações dos dados de entrada para que a saída seja coerente.

Esta classe de algoritmos é pouco interessante em um problema do planejamento de locomoção, pois seria necessário que um especialista, que nem sempre existe, definisse os movimentos mais indicados para um conjunto de posições para então se realizar a extrapolação para as demais.

4.1.2.
Aprendizado não supervisionado

No aprendizado não supervisionado o algoritmo extrai as características dos dados e os agrupa em classes (55). Geralmente, o aprendizado não supervisionado é aplicado em tarefas de agrupamento, que consistem em agrupar os dados em classes que contêm dados similares entre si, de acordo com alguma medida de similaridade. Este tipo de aprendizado possibilita que os dados sejam estudados mais cuidadosamente, pois a separação em grupos pode denotar semelhanças entre as amostras que não são necessariamente de fácil percepção apenas observando a base original.

Esta classe de algoritmos é razoavelmente interessante para o problema de planejamento de locomoção pois não necessita da definição da decisão a ser tomada. No entanto, não se conhece a priori uma amostra de dados capaz de representar todas as possíveis combinações dos dados de entrada, de forma que o aprendizado teria de ser em tempo real, em que a base de dados seria ampliada a cada interação. Esta limitação traz riscos para o equilíbrio do robô, pois o algoritmo pode designar grupos que não necessariamente requerem decisões distintas e, com isso, é muito difícil designar qual decisão cada grupo representa.

4.1.3.
Aprendizado por reforço

A classe de algoritmos que utilizam aprendizado por reforço apresenta, em geral, uma roupagem um pouco diferente das anteriormente citadas. O que seriam as entradas nas seções anteriores são agora denotados como estados. O conhecimento é proveniente de uma série de recompensas fornecidas pelo algoritmo para decisões que apresentem resultados positivos em prol da execução de uma tarefa desejada (56).

Como geralmente o algoritmo não tem conhecimento algum armazenado de início, é necessário testar comportamentos diversos para determinar quais ações devem ou não ser reforçadas.

A aplicação desta classe de algoritmos é bastante apropriada ao problema de planejamento de locomoção, pois todas as variáveis que o algoritmo necessita estão à disposição: as entradas são consideradas como estados e as ações serão os deslocamentos de cada junta. Assim, é possível gerar um padrão de locomoção e ao mesmo tempo desenvolver um método de detecção de perigo de queda de forma a penalizar as ações que possam provocar tal situação.

Para facilitar a compreensão do algoritmo é importante entender sua origem: o Processo de Decisão de Markov (MDP) (57).

4.1.3.1.
Processo de decisão de Markov (MDP)

Processos de decisão de Markov (MDPs), originariamente formulados em (57), fornecem uma estrutura matemática para a tomada de decisão em situações em que os resultados são parcialmente aleatórios e estão em parte sob o controle de um tomador de decisão. Um MDP é um problema de busca não determinística, ou seja, os resultados da tomada de decisão são incertos (58). Ele é definido por um conjunto de estados s que define um espaço de estados, um conjunto de ações a que define a forma como a tomada de decisão pode ser feita, uma função de transição $T(s, a, s')$ que representa a probabilidade de que uma ação a leve de um estado s para um estado s', e uma função de recompensa $R(s, a, s')$ que representa a recompensa que uma ação a merece por levar um estado s para um estado s'.

Um MDP deve apresentar um estado inicial, mas não necessariamente um estado final, de forma que é possível que funcione de forma perpétua caso necessário.

Para facilitar a compreensão das variáveis que compõem o algoritmo, utiliza-se o exemplo clássico ilustrado na Figura 25: uma espécie de labirinto onde um robô hipotético deve se deslocar pelos retângulos – a cor preta representa um obstáculo intransponível e as outras cores representam possíveis locais em que o robô pode se posicionar. Cada retângulo representa um estado em que o robô pode estar. Ele é colocado no estado INÍCIO e deve chegar ao OBJETIVO, tomando cuidado para não passar pelo estado de PUNIÇÃO. Ao atingir a PUNIÇÃO ou o OBJETIVO, o experimento é reiniciado e o robô é posicionado novamente no INÍCIO.

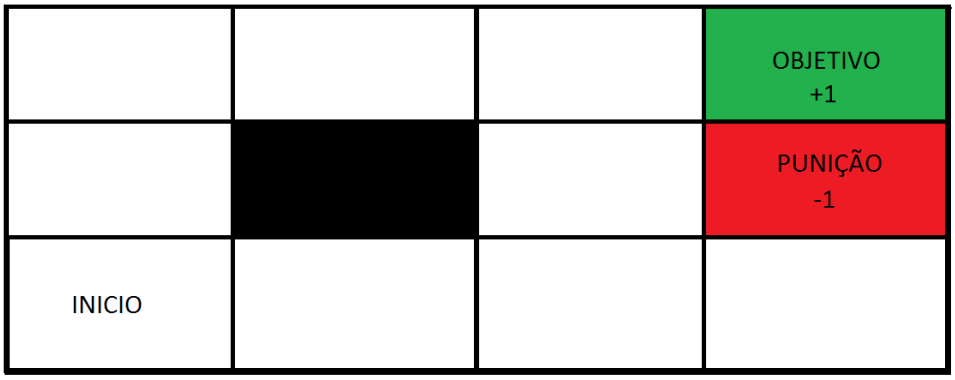

Figura 25 - Labirinto utilizado como exemplo.

O robô, denominado agente, ou seja, uma entidade ou programa capaz de realizar ações, é capaz de se mover nos quatro sentidos do labirinto, limitado apenas pelas paredes e obstáculos. Se o agente optar por se mover para uma das direções obstruídas, o robô fica parado e não executa ação alguma. Entretanto, existe um fator de incerteza atrelado à ação escolhida, de forma que, se o agente decidir mover-se para cima, por exemplo, existe uma probabilidade de este se deslocar em uma das direções adjacentes da escolhida. Neste caso, há uma probabilidade de 80% de o deslocamento se dar na direção desejada, e uma probabilidade de 10% para cada uma das adjacentes. Este fator de incerteza é representado pela função de transição $T(s, a, s')$.

São fornecidas pequenas recompensas para cada estado e grandes recompensas nos estados finais: +1 para o estado OBJETIVO e -1 para o estado

PUNIÇÃO. Estas recompensas são representadas pela função de recompensa $R(s, a, s')$.

Inicialmente, como o agente não tem conhecimento algum sobre o problema, selecionam-se ações de forma randômica. Com base nas probabilidades anteriormente estipuladas, a função de transição $T(s, a, s')$ determina, qual ação a realmente será executada. A função $R(s, a, s')$ recompensa ou pune esta ação a tomada para a transição do estado s para o estado s'.

Em problemas deste tipo, deseja-se obter uma espécie de planejamento ótimo ou uma sequência de ações que devem ser tomadas do início ao fim do problema. No caso de MDPs, deseja-se obter uma política ótima π^* (59). Uma política π permite que cada estado escolha uma ação, dentre todas as possíveis, a ser realizada. Uma política é dita ótima quando as ações que ela define, se seguidas, representam a maximização da pontuação obtida em cada estado.

A política ótima pode ser afetada por diversos fatores, mas principalmente pela definição das recompensas. A Figura 26 ilustra exemplos de políticas ótimas para diversas variações de recompensas para estados não terminais. Se definidas com valores negativos, existirá uma pressão para que se atinja logo um estado terminal. Se forem selecionados valores negativos maiores do que o estado final definido como PUNIÇÃO, o robô, agindo de acordo com uma política ótima, tende a realizar ações que levem ao estado final com recompensa positiva, denominado OBJETIVO. Quanto mais próxima de zero for a recompensa dos estados não terminais, menos risco o robô tende a correr pois o reforço negativo incentiva a busca por soluções mais seguras, mesmo que eventualmente mais longas, que aumentem a chance de obtenção de uma boa recompensa final. Caso a recompensa dos estados não terminais seja menor do que a oferecida no estado PUNIÇÃO, a política ótima faz com que o robô procure a saída mais rápida possível, que, no caso, é ser punido. Observe-se que valores positivos de recompensas nos estados não terminais não são interessantes para este tipo de problema, pois o robô poderia se mover de um lado para o outro eternamente e assim acumular o máximo de recompensa possível sem nunca atingir um dos estados finais.

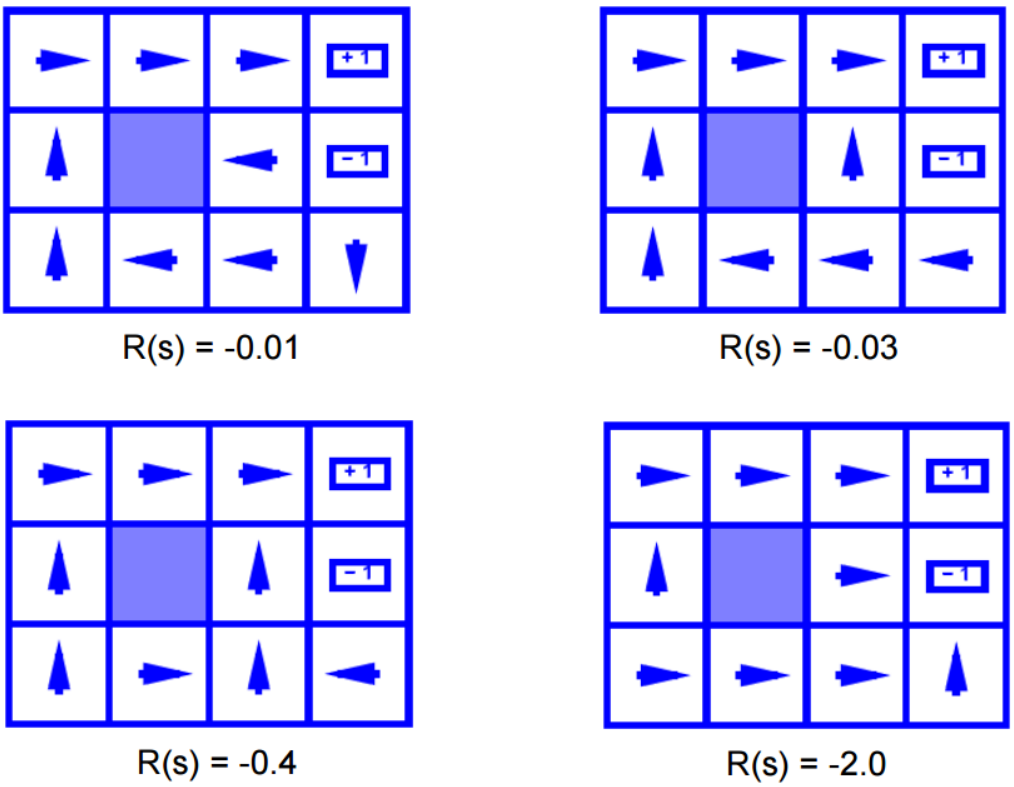

Figura 26 - Exemplos de políticas ótimas.

Executada uma ação, é necessário extrair o aprendizado adquirido por esta ação. Este aprendizado pode ser resumido em uma variável de recompensa r_n: a recompensa dada para a enésima transição de estado executada a partir do estado atual. A contabilização da pontuação da política é dada por uma função de utilidade genérica U que soma todas as recompensas geradas no percurso, funcionando como uma memória da política, quantificando o quão adequada ela seria (59). A função de utilidade aditiva pode ser representada pela seguinte equação:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots \quad (45)$$

Para que seja possível utilizar uma função de utilidade, supõe-se que o problema possui preferências estacionárias, ou seja, espera-se que a ação dita como ótima de um estado sempre seja ótima para este estado; somente assim faz sentido que a função de utilidade contabilize a pontuação da política com adições.

Com o uso da função de utilidade, surge um novo problema: quando se aborda um MDP sem estado final, a tendência seria a função de utilidade crescer indefinidamente, pois não há um termo final para a adição que compõe a função de utilidade. A solução para tal situação é utilizar uma taxa de desconto

exponencial γ, onde $0 < \gamma < 1$, que implementa uma espécie de decaimento das recompensas dos estados futuros. Assim, a função de utilidade tende a zero para recompensas muito à frente do estado atual. Além disso o aprendizado é melhorado, pois decisões mais imediatas e, portanto, comprovadamente mais importantes para o agente no estado atual, têm maior relevância. Com isto, a nova função de utilidade passa a ser uma adição descontada:

$$U([r_0, r_1, r_2, r_3, \dots]) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots \quad (46)$$

Como a próxima seção trata das equações de Bellman (60), é importante definir alguns conceitos.

Um estado s pode ser resumido por uma pontuação, denominada valor e caracterizada por uma função de utilidade $V(s)$. Um valor é definido como ótimo e, portanto, representado pela notação $V^*(s)$, quando, a partir do estado s, o agente age de acordo com uma política ótima. O par estado-ação também pode ser resumido por uma pontuação, denominado Q-estado, e é representado pela notação $Q(s,a)$. Ele é dito ótimo, e representado pela notação $Q^*(s,a)$, quando, a partir da ação a vinda do estado s, o agente age de acordo com uma política ótima dali em diante. A Figura 27 ilustra o processo completo de mudança de um estado s para um estado s'. É possível identificar diversos componentes presentes na estruturação de um MDP, como também onde os conceitos apresentados neste paragrafo se encaixam sequencialmente.

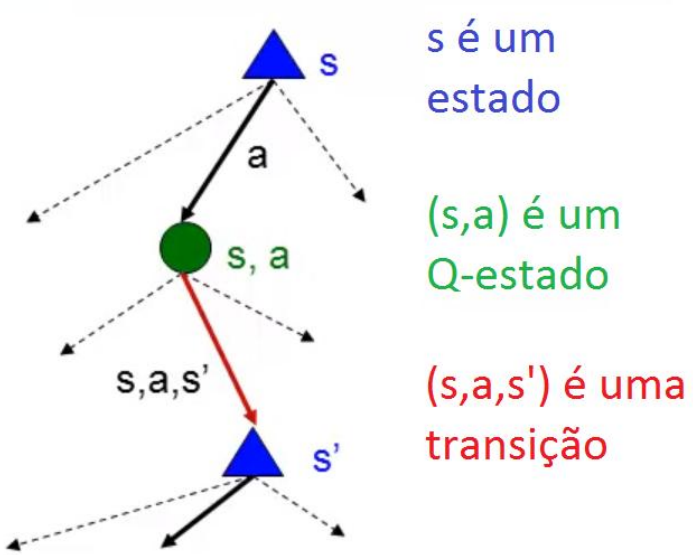

Figura 27 - Estrutura da mudança de estado.

4.1.3.1.1. Equações de Bellman

As equações de Bellman (60), base para a formulação do aprendizado por reforço, caracterizam como o valor ótimo $V^*(s)$ de um estado s pode ser calculado.

A pontuação de $V^*(s)$ pode ser facilmente entendida como a maior pontuação dentre os Q-estados ótimos $Q^*(s,a)$, já que, em uma politica ótima, a sequência de ações executadas é a que maximiza a pontuação do estado, ou seja:

$$V^*(s) = \max_a Q^*(s,a) \tag{47}$$

Como observado na Figura 27, $V^*(s')$ é obtida a partir da função de transição $T(s,a,s')$ originada de $Q^*(s,a)$. Como a função de transição é uma probabilidade, não é possível determinar que estado seria s'. Assim, determina-se uma soma do produto das probabilidades $T(s,a,s')$, com os valores ótimos $V^*(s')$ dos possíveis estados s'. Como um MDP é baseado em recompensas $R(s,a,s')$ em todos os estados, é necessário adicionar esta parcela ao valor ótimo do estado futuro. Por fim, pelo fato de o valor ótimo ser uma função de utilidade, faz-se uso de um fator de decaimento exponencial γ para valores de estados futuros, com o fim de evitar valores acumulados demasiadamente grandes. Assim, $Q^*(s,a)$ será dada por:

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{48}$$

e, a partir da Equação (47):

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{49}$$

Com este procedimento, é possível mensurar o valor de cada estado e assim utilizar uma política ótima.

Para se obter $V^*(s)$, é necessário conhecer $V^*(s')$, que depende de $V^*(s'')$ e assim por diante. Para resolver este problema, modifica-se a formulação de modo que o valor possa ser calculado de forma iterativa.

4.1.3.1.2. Iteração do valor

Para calcular de forma iterativa o valor $V(s)$ de um estado s consideram-se instantes de tempo discretos. Assim, o valor de $V(s)$ no instante discreto $k+1$ será função do seu valor no instante k (59):

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')] \quad (50)$$

onde $V_0(s) = 0$, de forma a garantir que no início do aprendizado a soma das recompensas seja zero (nenhuma ação é privilegiada na fase inicial de aprendizado).

Voltando ao caso do exemplo do robô no labirinto, uma possível política π poderia escolher ações de forma aleatória, de forma a explorar e aprender novos valores para os estados. A cada ação executada, uma nova pontuação para o valor do estado, onde esta ação foi gerada, é calculado pela Equação (50) e o processo se repete até que um dos estágios finais seja atingido e o robô é posicionado novamente no início. Após algumas poucas repetições deste procedimento é possível modificar a política e passar a escolher as ações que levam a estados com maior valor, o que se constituiria em uma política ótima π^*.

4.1.3.2. Aprendizado baseado em modelo

O algoritmo da seção anterior é capaz de resolver o problema do planejamento da locomoção mas requer variáveis que são muito difíceis de mensurar em uma aplicação real, como, por exemplo, as probabilidades que compõem as funções de transição $T(s,a,s')$ e as recompensas das funções $R(s,a,s')$. Tais funções podem ser modeladas empiricamente, de forma a obter, após um número suficiente de repetições, um modelo do problema a ser utilizado pelo MDP. Esta metodologia é denominada aprendizado baseado em modelo (59). Entretanto, em aplicações que requerem aprendizado em tempo real, o processo de estimativa das variáveis requer muitas iterações e, por não ser robusto, pode ocasionar quedas do modelo robótico. O ideal seria a utilização de um aprendizado sem conhecimento a priori de $T(s,a,s')$ e $R(s,a,s')$.

4.1.3.3. Aprendizado livre de modelo

A ideia base de um aprendizado livre de modelo é tentar estimar o que for importante, como, por exemplo, os valores dos estados ou os Q-estados, de uma forma robusta o suficiente para compensar o desconhecimento das variáveis $T(s,a,s')$ e $R(s,a,s')$ (59).

Considere-se um caso simples de um ambiente de aprendizado por reforço onde a ação a do estado s é determinada por uma política $\pi(s)$ e o objetivo é aprender os valores de cada estado sem nenhum conhecimento relativo a $T(s,\pi(s),s')$ e $R(s,\pi(s),s')$. O aprendizado deve ser realizado em tempo real e de forma passiva, ou seja, sem escolha de que ação será tomada; será executada apenas a política e o aprendizado será feito pela experiência adquirida. Como não é mais necessário considerar todas as ações, a equação para cálculo do valor do estado é:

$$V_{k+1}^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')] \quad (51)$$

onde V_k^{π} representa o valor do estado seguindo a política π na iteração k.

Para determinar a recompensa $R(s,\pi(s),s')$, é importante entender o desempenho desejado para o agente, como visto na Figura 26 e explicado anteriormente. A definição das recompensas finais e intermediárias varia muito de acordo com o nível de resposta desejado do agente. Se for desejada uma resposta que minimize a quantidade de estados intermediários, é interessante que tais estados tenham uma pequena punição, estimulando assim que um estado final de recompensa seja encontrado em poucas transições, maximizando o somatório das recompensas. Se for de interesse minimizar a probabilidade de que sejam interceptados estados finais de punição, é indicado não dar reforços intermediários ou dar punições muito pequenas, para estimular o agente a seguir caminhos mais seguros, mesmo que mais transições de estados sejam necessárias. Estes exemplos servem para demonstrar que a definição das recompensas pode variar para cada aplicação, mas, de forma básica, é necessário definir apenas as recompensas dadas para os estados finais: como positivas, no caso de se atingir um estado objetivo, ou negativas, como no caso de se atingir um estado de punição.

Em relação às probabilidades de transição $T(s, \pi(s), s')$, na Equação (50) foi determinado que o valor atualizado do estado é composto por um somatório dos produtos das probabilidades de transição $T(s, \pi(s), s')$ entre dois possíveis estados e um termo que contem a recompensa $R(s, \pi(s), s')$ e a estimativa do valor do estado futuro $V^{\pi}(s')$. Em um aprendizado livre de modelo, que se baseia no desconhecimento das probabilidades de transição $T(s, \pi(s), s')$, pode-se estimar esta parcela pela aquisição de diversas amostras de execuções de transições do estado s para o estado s' seguindo a política $\pi(s)$. Assim, a transição é estimada pela prática e ponderada, obtendo-se uma boa estimativa para as probabilidades. Isto pode ser descrito pelas seguintes equações:

$$
\begin{aligned}
amostra_1 &= R(s, \pi(s), s_1') + \gamma V_k^{\pi}(s_1') \\
amostra_2 &= R(s, \pi(s), s_2') + \gamma V_k^{\pi}(s_2') \\
&\ldots \\
amostra_n &= R(s, \pi(s), s_n') + \gamma V_k^{\pi}(s_n') \\
V_{k+1}^{\pi}(s) &= \frac{1}{n} \sum_{i=1}^{n} amostra_i
\end{aligned}
\tag{52}
$$

onde n corresponde ao número de amostras obtido.

Tais aquisições não são possíveis em muitas aplicações reais, já que, ao se executar uma ação que muda o estado, nem sempre é possível retornar exatamente ao estado anterior. Assim, utiliza-se uma forma adaptada desta ideia, denominada aprendizado por diferenças temporais.

4.1.3.3.1. Aprendizado por diferenças temporais

O aprendizado por diferenças temporais (61) leva em conta a possível impossibilidade de retorno a estados anteriores e, por isso, utiliza-se apenas a amostra obtida na última atuação que resultou na transição do estado s para o estado s', acrescida de uma variável α, onde $0 \leq \alpha \leq 1$, que define uma proporção de relevância do valor do estado atual e da última amostra, implementando assim uma taxa de aprendizado (56). Por se tratar de um processo iterativo, a estimativa do valor de um estado tende a melhorar a cada vez que o agente o visita. A variável α implementa uma média móvel exponencial, pois os

valores amostrais muito antigos tendem a ter a sua importância diminuída frente aos novos valores amostrais mais representativos do estado. A equação tem a seguinte forma:

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)amostra$$
$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha[amostra - V^{\pi}(s)] \quad (53)$$

ou seja:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha[R(s,\pi(s),s') + \gamma V^{\pi}(s') - V^{\pi}(s)] \quad (54)$$

Com essa equação torna-se possível calcular os valores de cada estado de acordo com uma política π, mas não se pode assegurar que ela é ótima. Torna-se necessário, portanto, determinar uma forma de se abstrair da política e selecionar as ações que sejam capazes de realizar o objetivo proposto. Em uma aplicação genérica, é interessante utilizar os Q-estados $Q(s,a)$, definidos por um par estado-ação, no processo de aprendizado.

4.1.3.3.2. Q-Learning

Para formular o algoritmo denominado de *Q-Learning* (59), é necessário reescrever as equações utilizadas para calcular os valores dos estados em função dos valores dos Q-estados. Recorde-se que:

$$Q(s,a) = \sum_{s\prime} T(s,a,s')[R(s,a,s') + \gamma V(s')] \quad (55)$$

Para que a função dependa apenas dos valores dos Q-estados é necessário substituir a variável $V(s')$ por alguma variável conhecida. Com base na Equação (47), é possível escrever:

$$se: V(s) = \max_{a} Q(s,a)$$
$$entao: V(s') = \max_{a\prime} Q(s',a') \quad (56)$$

com isto, a Equação (55) fica:

$$Q(s,a) = \sum_{s\prime} T(s,a,s')[R(s,a,s') + \gamma \max_{a\prime} Q(s',a')] \quad (57)$$

utilizando a Equação (50) tem-se:

$$Q_{k+1}(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right]$$
$$ou$$
$$Q(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \quad (58)$$

ou, a partir do desenvolvimento na seção anterior:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (59)$$

onde $Q_0(s,a) = 0$, de forma a garantir que no início do aprendizado a soma das recompensas seja zero.

Diferentemente do aprendizado baseado nos valores dos estados, que dependem de uma política ótima π^* para tomar ações ótimas, o algoritmo *Q-Learning* permite que, sem conhecimento prévio das probabilidades de transição $T(s,a,s')$, das recompensas $R(s,a,s')$ e da política ótima π^*, e com exploração suficiente do espaço de estados, seja possível convergir em todas as ações que pertençam à política ótima, mesmo que durante a exploração o agente tenha agido de maneira sub-ótima (59). É recomendado utilizar uma taxa de aprendizado α que seja decrescente com o número de iterações até um ponto, onde seja baixa o suficiente. A taxa de aprendizado não deve decrescer muito rapidamente, pois, como nas primeiras iterações há muita variabilidade dos valores dos Q-estados, é necessário reduzi-la de forma cautelosa para não forçar à estabilidade Q-estados que ainda não haviam se aproximado do valor que seria realmente adequado.

Não é recomendada a realização de ações de forma totalmente randômica pois a exploração, pelo agente, de todo o espaço de estado pode levar muito tempo. No caso de ações que sejam danosas ou destrutivas ao agente, serão utilizados mais agentes do que necessário.

4.1.3.3.2.1. Exploração (descobrir) vs. Exploração (utilizar)

Quando o agente não conhece nada sobre o estado, é necessário que ele explore as possíveis ações para obter novos aprendizados e assim mensurar os

valores que possam envolver aquele estado. Por outro lado, quando um agente já conhece um conjunto de ações que leve ao objetivo, mesmo sem saber se tais ações são as ótimas, ele pode explorar (no sentido de utilizar) este conhecimento, de forma a atingir o objetivo. Para atingir um equilíbrio entre estas duas faces do algoritmo, desenvolveram-se várias abordagens para balancear a exploração (descoberta) dos estados (62, 63, 64). Uma das mais utilizadas e consideravelmente simples é a ε-*greedy*, na qual uma variável ε, onde $0 \leq \varepsilon \leq 1$, representa uma porcentagem. Tal valor é geralmente pequeno e determina a proporção ε em que a ação deve ser tomada de forma aleatória e a proporção $(1 - \varepsilon)$ em que deve ser escolhida a ação com maior valor de Q-estado.

Essa solução permite uma redução do tempo de exploração (descoberta) do espaço de estados, mas ainda não garante que tal exploração seja realmente necessária, pois as ações ainda são aleatórias. Para solucionar tal problema, adiciona-se uma função de exploração (59), que complementa a tomada de decisão aleatória. Uma forma simples é:

$$f(u, n) = u + \frac{k}{n} \tag{60}$$

onde u seria o valor do Q-estado, n um contador do número de visitas a tal Q-estado e k é um valor fixo alto o suficiente para ser relevante na ocasião da escolha do próximo Q-estado. A Equação (59) passa a ser, portanto:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\{R(s, a, s') + \gamma \max_{a'} f[Q(s', a'), N(s', a')] - Q(s, a)\} \tag{61}$$

onde $N(s', a')$ representa um contador do número de visitas a $Q(s', a')$. Quanto menos visitas $Q(s', a')$ tiver, maior o estimulo de visitá-lo, pois maior será a relevância da variável k que eleva artificialmente o valor do Q-estado. Na situação oposta, quanto maior o número de visitas, menor será a parcela que contém o termo k; assim,assim a função aproxima-se do valor original do Q-estado.

É muito comum, na robótica, definir os estados de um aprendizado por reforço como possíveis posições em que o agente pode estar, seja por meio de coordenadas – caso do exemplo do robô móvel da Figura 25 – ou por ângulos das juntas – caso de braços robóticos. No exemplo do robô móvel, ao se variar o tamanho do labirinto, mantendo-se os estados finais e iniciais nas mesmas posições relativas, aumenta-se o tamanho do espaço de estados do agente, e com

isto o aprendizado se torna mais demorado, pois há a necessidade de visitar boa parte destes estados até que seja encontrada um trajeto viável para o agente. Além disso, é necessário repetir o problema algumas vezes para que um reforço positivo do estado final repercuta no estado inicial, fazendo assim com que o agente se mova desde o início em direção ao objetivo, como desejado. Para solucionar o problema relacionado ao aumento do tempo de aprendizado causado pela dimensão do espaço de estados, faz-se uso de uma metodologia capaz de generalizar estados que tenham comportamentos similares e de quantificar, de forma aproximada, os valores destes estados. É necessário, inicialmente, utilizar uma representação dos estados não como simples posições, mas como características.

4.1.3.3.2.2. Representação baseada em características

Nesta forma de representação dos estados, são observadas algumas propriedades de cada posição, que geralmente devem ser representadas entre zero e um (65). No caso do exemplo do robô móvel, um estado poderia ser resumido por direções cardeais até a posição do objetivo, até a posição da punição mais próximo e até a posição do obstáculo mais próximo.

Este tipo de representação também pode ser usada nos Q-estados. No caso do exemplo, uma variável binária poderia identificar um Q-estado como aproximador do objetivo (valor um), ou afastador do objetivo (valor zero). Existem inúmeras formas de representar estados e Q-estados; as mencionadas são apenas exemplos.

É possível, agora, identificar características de cada estado e, assim, agir de forma similar para os que se assemelham. Para isso é necessário adaptar a forma como os valores dos estados e Q-estados são computados.

4.1.3.3.2.3. Funções de valor linear

Utilizando a representação baseada em características, que em geral deve estar normalizada, faz-se uma soma ponderada por pesos w com o objetivo de

formar uma função linear de n dimensões; esta função passa a contabilizar os novos valores para os estado e Q-estados (65):

$$V(s) = w_0 + w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s)$$
$$Q(s,a) = w_0 + w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a) \quad (62)$$

Este tipo de formulação tem a vantagem de representar diversos estados similares por algumas características comuns. É importante ressaltar, contudo, que uma escolha de características que representem mal estados totalmente distintos – não representar a proximidade de punições, por exemplo –, pode permitir que valores baixos sejam atribuídos a estados que não o merecem.

Resta definir os valores dos pesos w. A solução baseia-se no mesmo princípio da Equação (61) do *Q-Learning*.

4.1.3.3.2.4. *Q-Learning* Aproximado

Para atualizar os pesos, é necessário realizar uma simples minimização de erros (65), ou seja, computar a derivada do erro de cada peso e subtrair o resultado multiplicado por uma taxa de aprendizado α para controlar o ajuste:

$$w_i \leftarrow w_i - \alpha \frac{\partial\, erro(w)}{\partial w_i} \quad (63)$$

A Equação (61) é composta de duas parcelas no seu lado direito: a primeira é o próprio Q-estado e a segunda é uma multiplicação da taxa de aprendizado α pelo valor atualizado do Q-estado menos o valor anterior do Q-estado, configurando assim um termo de erro. Tal afirmação é mais facilmente visualizada abaixo:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{erro\} \quad (64)$$

onde:

$$erro = Q_{novo} - Q_{velho} = [R(s,a,s') + \gamma \max_{a'} f[Q(s',a'), N(s',a')] - Q(s,a) \quad (65)$$

prosseguindo:

$$erro(w) = \frac{1}{2}[Q_{novo}(w') - Q_{velho}(w)]^2 \quad (66)$$

$$\frac{\partial\, erro(w)}{\partial w_i} = -[Q_{novo}(w') - Q_{velho}(w)]f_i(s,a)$$

A derivada só é possível, pois $Q_{velho}(w)$ é função de w_i segundo a Equação (62), e $Q_{novo}(w')$ não é função da variável derivada, pois ele depende de pesos w' de Q-estados futuros. Com isso a equação final para o *Q-Learning* aproximado é:

$$w_i \leftarrow w_i + \alpha\{[R(s,a,s') + \gamma \max_{a'} f[Q(s',a'), N(s',a')] - Q(s,a)\}f_i(s,a) \qquad (67)$$

Com os conhecimentos apresentados até este ponto é possível resolver o problema de planejamento de locomoção sem a necessidade de muito poder computacional. Como o aprendizado por reforço abre possibilidades para muitas configurações de suas variáveis, nas seções seguintes serão definidas as abordagens para cada elemento que compõe o aprendizado, de forma a possibilitar um aprendizado fluido e rápido do robô.

4.2. Definição das variáveis do problema

Uma caminhada bípede é composta por duas parcelas de equilíbrio utilizando as duas pernas, e duas parcelas de equilíbrio com uma perna só. Todas as parcelas podem ser resumidas no deslocamento ou manutenção do centro de massa do robô em uma determinada região do polígono de suporte. A primeira etapa no processo de solução é definir o espaço de estados em que o robô exercerá ações.

4.2.1. Espaço de estados

Um possível espaço de estados seria definido pela posição, em relação ao eixo de referência fixo, do ponto no solo, no plano transverso, que representa a projeção das coordenadas x e y do centro de massa. Isto determinaria estados como cada conjunto de coordenadas, conforme representado na Figura 28, onde é possível visualizar, à esquerda, o polígono de suporte, e à direita, um exemplo de uso do espaço de estado proposto.

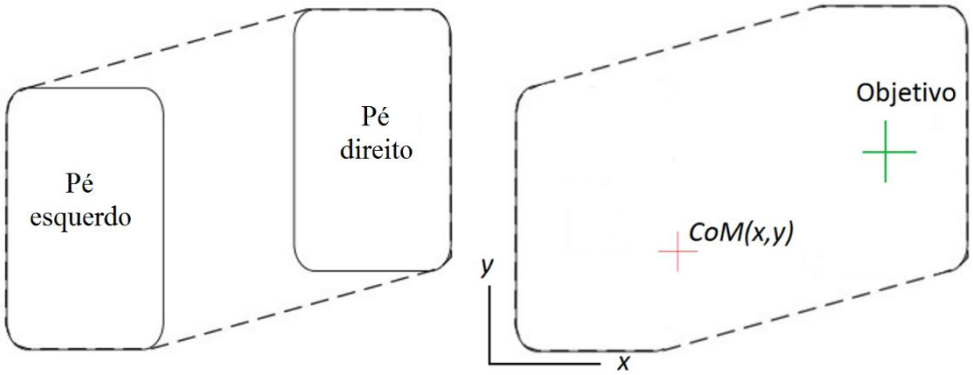

Figura 28 - Exemplo de uso da primeira proposta de espaço de estados.

Esta abordagem, apesar de muito similar à clássica e, portanto, com maior probabilidade de êxito, sofre muito com a variação da precisão do ponto do centro de massa, pois, quanto mais casas decimais forem utilizadas, maior é o espaço de estados.

Em uma outra abordagem, o espaço de estados é determinado por direções: uma direção para o objetivo e outra para a penalização mais próxima. Assim, o espaço de estados é composto por aproximadamente sessenta e seis estados, onde sessenta e quatro seriam todas as combinações das oito possíveis direções (norte, nordeste, leste, sudeste, sul, sudoeste, oeste e noroeste) para o objetivo e penalização mais próxima, e dois para sinalizar que foi atingido, com uma pequena tolerância, um objetivo ou uma penalização. A Figura 29 demonstra a aplicação do espaço de estado proposto.

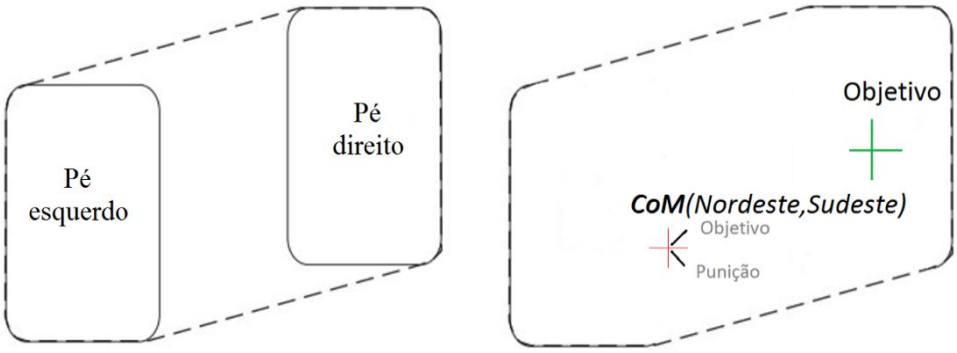

Figura 29 - Exemplo de uso da representação de espaço de estados utilizada.

Com esta representação é possível utilizar um número fixo de estados pois, independentemente da precisão do centro de massa e do objetivo, é plausível definir a direção que deve ser seguida e a que deve ser evitada em um determinado momento por uma simples diferença entre as coordenadas do

objetivo e do centro de massa, sempre em relação aos sistemas de coordenadas fixo da base.

Vale ressaltar que o espaço de estados é sempre o mesmo, independentemente da fase de equilíbrio. A alteração na forma do espaço de estados só reduz a mobilidade do centro de massa

4.2.2. Recompensas

Nos estados que representam a chegada na região de tolerância do objetivo e da punição, serão dadas, respectivamente, uma recompensa positiva (+20), e uma negativa (-10). Como, para os estados intermediários, o objetivo era obter as soluções que necessitassem de poucos passos, utiliza-se uma pequena recompensa negativa (-0.001), para evitar que soluções boas, porém muito próximas as áreas de punição sejam obtidas. É possível ver na Figura 30, à direita, a superfície do polígono de suporte. Toda vez que uma ação leva um estado para a área vermelha, que sinaliza a proximidade da extremidade do polígono de suporte, a ação é penalizada com o estado de punição e o agente é obrigado a executar a ação que o levou àquela posição no sentido reverso, com o intuito de deixar aquele lugar. Caso o agente atinja a área verde claro, o agente é premiado com a recompensa do objetivo e termina sua função.

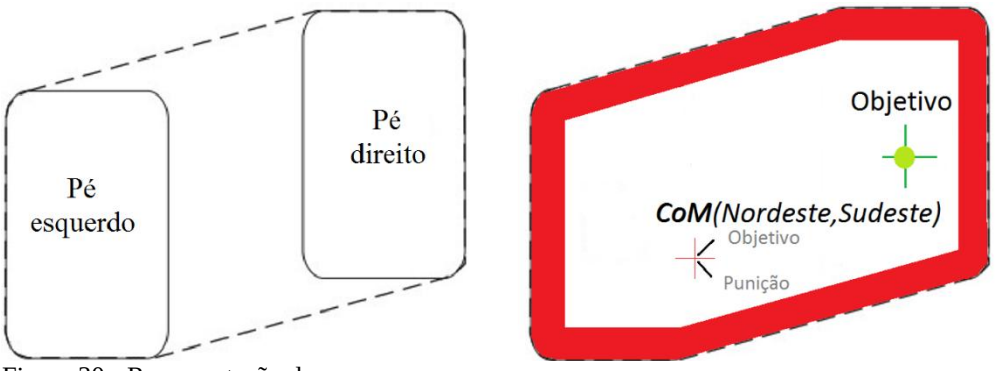

Figura 30 - Representação das recompensas.

4.2.3. Ações

As ações podem compreender oito possíveis movimentos: norte, nordeste, leste, sudeste, sul, sudoeste, oeste e noroeste. O movimento se dá ao se

incrementar ou decrementar por um valor pequeno o ângulo atual de um conjunto de juntas (0,01 graus neste trabalho).

Na fase de equilíbrio bípede, em que ambos os pés estão exercendo força, é necessário ter cuidado com a seleção das juntas pois, se for retirada a sustentação de um dos pés, o robô tem grandes chances de cair. Assim, seleciona-se um conjunto de juntas nos quadris e outro nos calcanhares de forma a manter os pés sempre paralelos ao chão. Tal combinação de juntas é utilizado para mover o centro de massa no plano frontal. Para movê-lo no plano sagital, inclina-se o torso para frente e para trás. A Figura 31 ilustra onde estão localizadas as juntas em questão, sinalizadas pelas cores verde e vermelha, respectivamente.

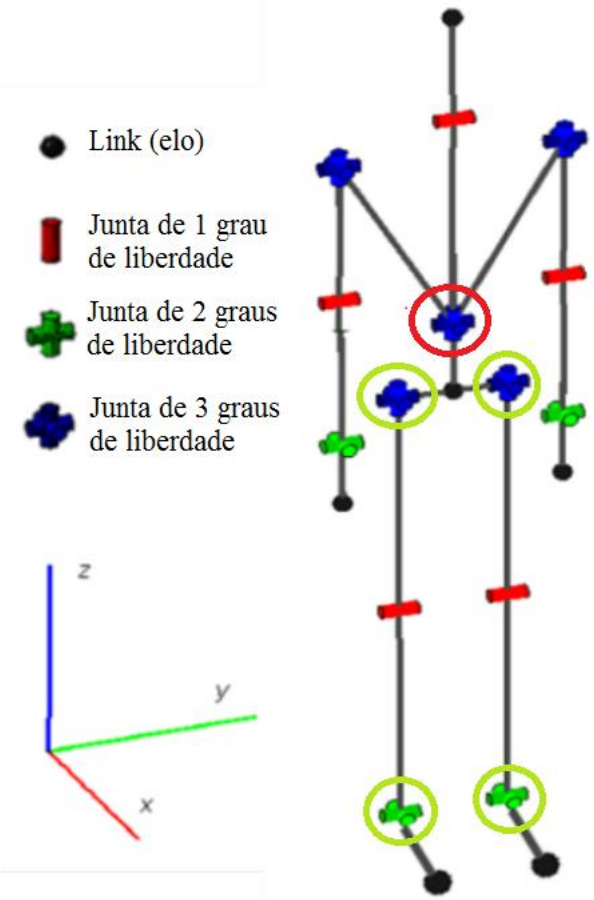

Figura 31 - Localização das juntas selecionadas para ação na fase de equilíbrio bípede.

Na fase de equilíbrio com um pé, há uma maior flexibilidade para seleção das juntas, pois não há mais restrições causadas por um apoio duplo. Desta forma é plausível utilizar apenas o conjunto circulado de verde na Figura 31. A parcela das juntas que pertence ao pé de apoio pode mover o centro de massa para todas as oito direções; a que não está apoiada pode manter a perna alinhada com a

apoiada e assim evitar que sejam tentadas posições em que as duas pernas possam eventualmente colidir.

Como a ação é incrementar um ângulo, não há como saber, em tempo real, se a ação resultará em uma posição das juntas que esteja fora do polígono de suporte. Como há uma diferença temporal entre o momento do envio do comando de posicionar as juntas em um determinado conjunto de ângulos e o instante em que a ação é executada, é possível que o ângulo seja incrementado de forma excessiva, levando a uma queda do robô. Nesta situação, é tarde demais para decrementar o ângulo, pois somente após alguns milissegundos o ângulo efetivado é detectado pelos sensores como em um estado de punição. Para contornar tal problema, utiliza-se a cinemática direta dos ângulos do robô para computar a posição futura do centro de massa com os ângulos desejados e assim evitar incrementos de ângulo excessivos, assegurando que a ação não ultrapasse a região de punição assinalada pela área vermelha na Figura 30.

Como existem oito ações possíveis e sessenta e quatro estados em que é possível escolher ações, existe um total de duzentos e cinquenta e seis Q-estados para este problema, independentemente da precisão da posição da junta. Com isto, há a possibilidade de utilizar o agente da forma mais precisa possível, limitada apenas pela precisão dos sensores e atuadores.

4.2.4. Características representadas

Foram selecionadas cinco características a ser ponderadas pelo *Q-Learning* Aproximado: um par binário representando se o deslocamento no plano sagital está no sentido correto em relação ao objetivo e o valor oposto ao primeiro; outro par binário representando se o deslocamento no plano frontal está no sentido correto em relação ao objetivo e o valor oposto ao primeiro; um binário que sinaliza se a ação selecionada levou a uma punição.

4.2.5. Agentes

Para finalizar, são definidos formalmente os tipos de agentes que constituem a locomoção. Inicia-se com o tipo de agente responsável pelo deslocamento do

centro de massa na fase de equilíbrio bípede – mais especificamente, o deslocamento é direcionado ao centro do pé mais à frente em relação ao outro; se os dois estiverem igualmente a frente, o pé direto é o objetivo. O segundo tipo de agente é responsável pela manutenção do centro de massa na área de tolerância do objetivo do agente anterior. Graças a ele é possível retirar o outro pé do chão e executar o deslocamento deste para frente. Este movimento é feito pelo simples deslocamento das juntas do joelho e do pé, que permitem que a perna seja retirada do chão e deslocada para frente, quando é baixada novamente, fazendo contato com o chão. Após este contato, o processo é passado para o primeiro agente, que reposiciona o centro de massa novamente.

Este capitulo tratou dos conceitos que permitiram o desenvolvimento dos agentes que determinam a trajetória do centro de massa a fim de possibilitar a locomoção de um robô bípede.

5.
Desenvolvimento e Resultados

Este capítulo tem por objetivo apresentar e avaliar o desempenho de diferentes abordagens de treinamento para o aprendizado do planejamento de locomoção proposto. São utilizados três agentes, um atuante na fase de suporte com duas pernas e dois na fase de apoio com uma perna – um relativo a cada perna. A Figura 32 apresenta o fluxograma correspondente à interação entre os agentes no ciclo de locomoção proposto.

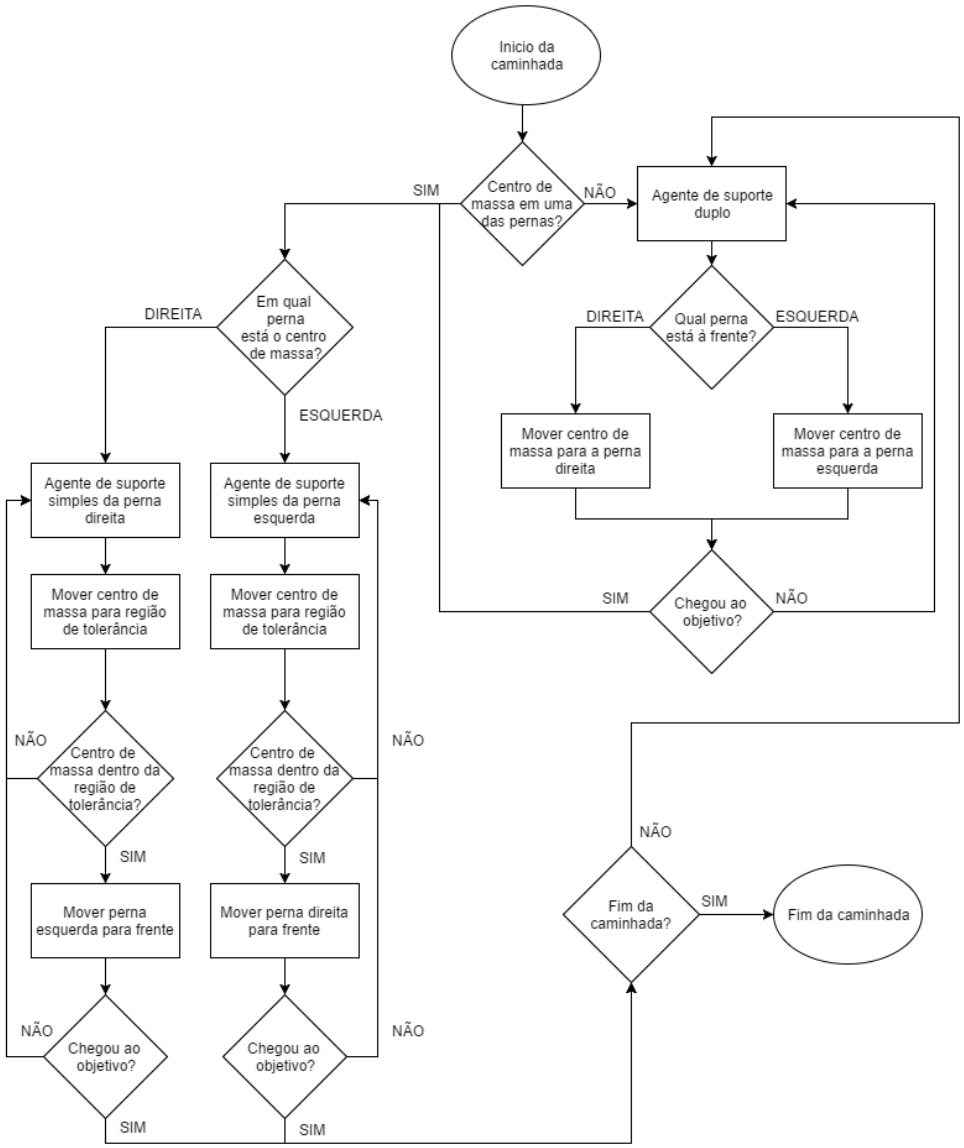

Figura 32 - Fluxograma da interação entre os agentes.

Selecionou-se, inicialmente, o agente responsável pelo deslocamento do centro de massa na fase de apoio das duas pernas no solo. Foram realizados, então, dois diferentes testes em escala para avaliar a velocidade de aquisição do conhecimento no algoritmo de aprendizado por reforço selecionado:

- Aplicação do algoritmo diretamente no ambiente em escala: aprendizado de forma clássica, em tempo real.
- Período de aprendizado em um ambiente simulado reduzido e posterior aplicação ao ambiente em escala: aprendizado em um ambiente simulado bem menor do que o ambiente de teste, de forma a obter noções básicas de direcionamento; a cada êxito do agente, a área do ambiente simulado aumenta gradualmente até que não mais ocorram mudanças significativas nos pesos do *Q-Learning* aproximado. Neste momento ele é inserido no ambiente de teste.

Na Figura 33 é possível verificar o desempenho em um ambiente simulado em escala do polígono de suporte. A área de mobilidade é um quadrado de lado correspondente a dez unidades de deslocamento. O gráfico é função de uma fração que representa a quantidade de deslocamentos que o agente teve que realizar (para cada iteração do algoritmo) para atingir o objetivo, dividida pela quantidade de deslocamentos dita ótima -. Foram executadas vinte repetições do experimento para cada uma das abordagens sugeridas. Os dados são representados de forma estatística, em que a parte do meio, vermelha, representa a mediana dos dados, a parte inferior da caixa, o percentil 25%, a parte superior, o percentil 75%, o limite inferior da linha pontilhada é o menor valor dos dados sem *outliers*; na parte superior, o maior valor também não considera *outliers*.

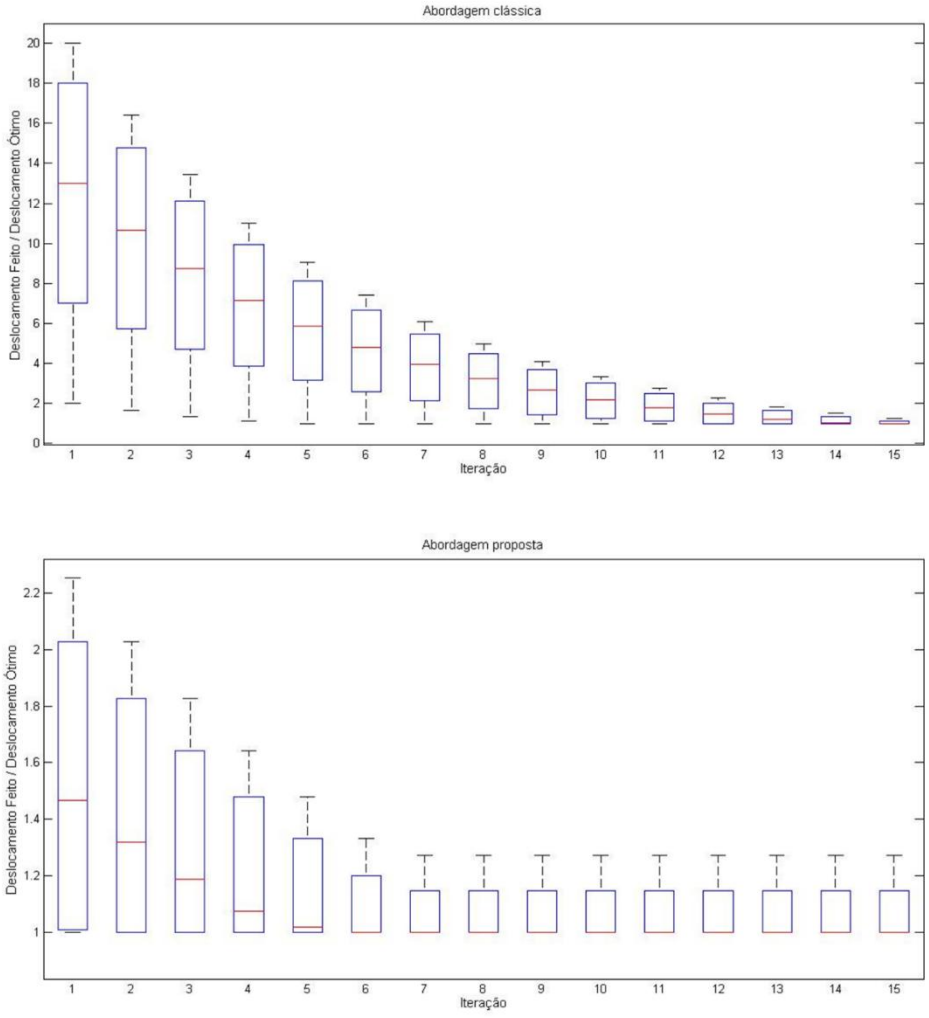

Figura 33 - Resultados dos testes em ambiente simulado em escala.

O gráfico superior representa o desempenho do aprendizado quando o algoritmo é aplicado diretamente ao problema; o gráfico inferior retrata o desempenho com treinamento prévio e aumento gradual do espaço de atuação, partindo de um quadrado com lado correspondente a três unidades de deslocamento. No caso deste experimento, o tamanho do lado é aumentado de três unidades assim que o agente consegue se deslocar até o objetivo, ao menos uma vez, para diversas posições iniciais e localizações de objetivos. Este treinamento prévio consumiu um minuto, por conta de um atraso adicionado ao código para tornar possível observar a locomoção do ponto de teste. Retirando-se a parte gráfica, o treinamento prévio tomaria alguns segundos. A abordagem clássica necessitou de cerca de três minutos para aprender o percurso. Em consequência, a abordagem aqui proposta mostrou-se mais eficiente e, portanto, mais indicada para a aplicação real.

Analisando-se os gráficos, também é possível concluir que a abordagem que se utiliza de ambientes de treinamento reduzidos apresenta melhores resultados, pois o agente já chega ao ambiente simulado com uma noção básica da ação que deve ser tomada, cabendo apenas ajustar de forma refinada os pesos do *Q-Learning* aproximado.

É possível avaliar, na Figura 34, a evolução de uma das trajetórias, descrita pelo centro de massa no modelo robótico, em que o agente utiliza a metodologia proposta. A parte superior da figura apresenta imagens que demonstram o posicionamento do corpo como um todo; a parte inferior concentra-se apenas nos pés. A parte inferior esquerda mostra um segmento em vermelho representativo da primeira iteração no modelo do robô após o período de treino em ambiente reduzido; o segmento em azul corresponde ao melhor trajeto possível, denominado trajeto ótimo. A figura da direta obedece à mesma regra das cores da figura da esquerda, mas o segmento vermelho representa a trajetória descrita após dez iterações da trajetória. Nota-se que há um refino da trajetória, mas sem atingir a trajetória ótima. Isto se justifica pelo fato de o espaço de estados do agente ser significativamente reduzido, forçando assim que diversos estados sejam resumidos como um e, em consequência, atrelados a uma ação dita ótima. Entretanto, o tempo de treinamento foi consideravelmente mais baixo, possibilitando assim que o agente fosse treinado rapidamente em um ambiente reduzido e aplicado em ambiente com aprendizado em tempo real sem grandes riscos à integridade estrutural do robô.

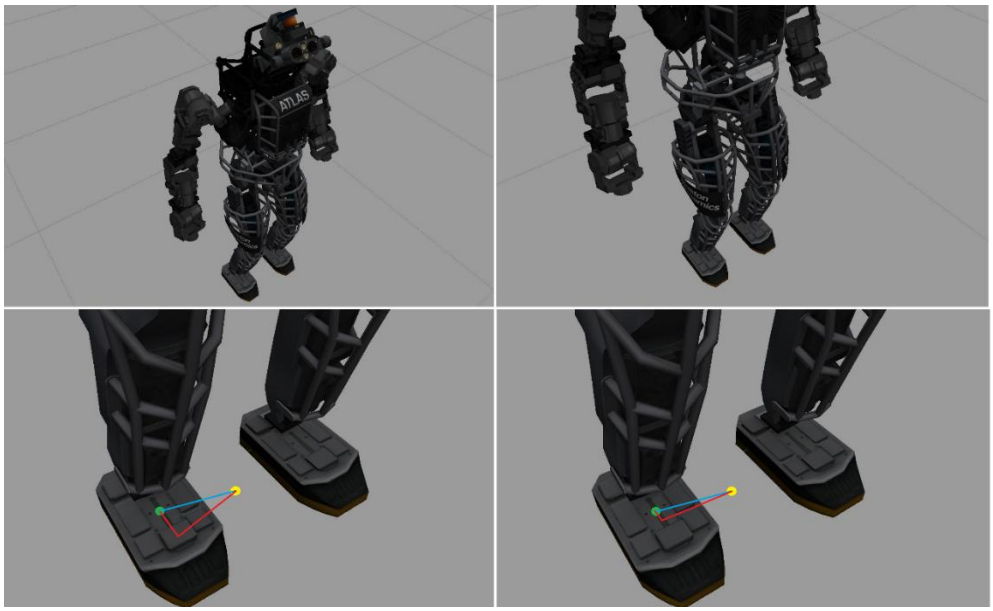

Figura 34 - Modelo robótico e evolução da trajetória em tempo real.

Com o agente responsável pela fase de apoio duplo das pernas treinado, é necessário desenvolver o agente responsável pela fase de apoio com uma perna. Como o sistema é de locomoção estática, não houve grandes preocupações com possíveis perturbações decorrentes da retirada do pé, que não é de apoio, do chão, já que esta operação é lenta. A Figura 35, apresenta uma sequência de imagens que retratam a trajetória descrita pela perna livre: desde o momento em que é retirada do chão até o momento em que é novamente apoiada. Vale ressaltar que esta trajetória não é determinada por um agente, mas sim calculada de forma que a perna levante, se mova para frente e abaixe, sempre da mesma forma. Tal escolha não foi atribuída a um agente pois o processo de treinamento requereria muitos estados para mapear a posição, com precisão relativamente alta, de todas as juntas da perna, tornando o aprendizado muito demorado. Haveria também a possiblidade de o agente tentar ações que resultassem em uma desestabilização do robô – fazer força no chão ou repousar o pé de forma irregular, por exemplo.

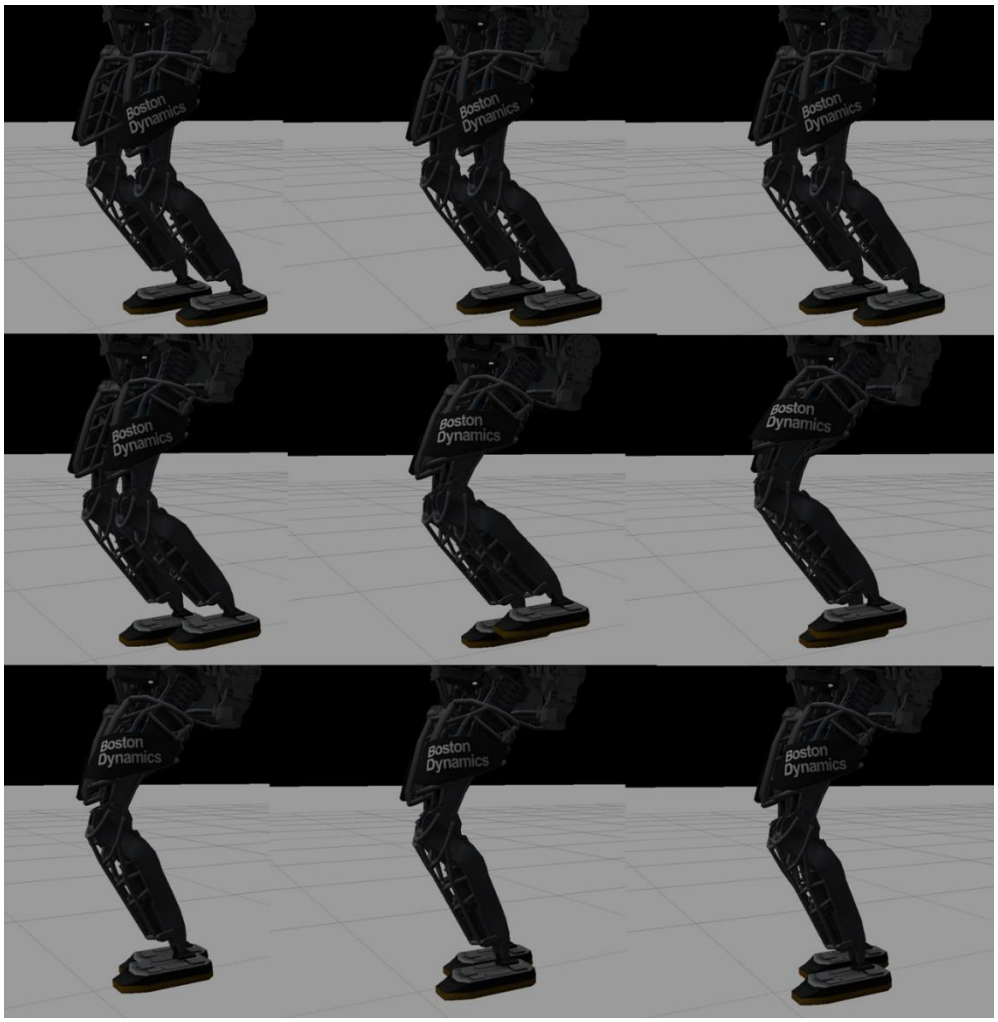

Figura 35 - Movimento da perna direita livre de restrições.

As Figura 36 e Figura 37 mostram um ciclo de locomoção completo do robô, desde o deslocamento do centro de massa na fase com apoio duplo, passando pela fase de apoio com uma perna só, em que o pé sem sustentação é retirado do chão, até a recolocação do pé mais à frente no chão; as fases se repetem, mas desta vez com o outro pé, e por fim o ciclo termina.

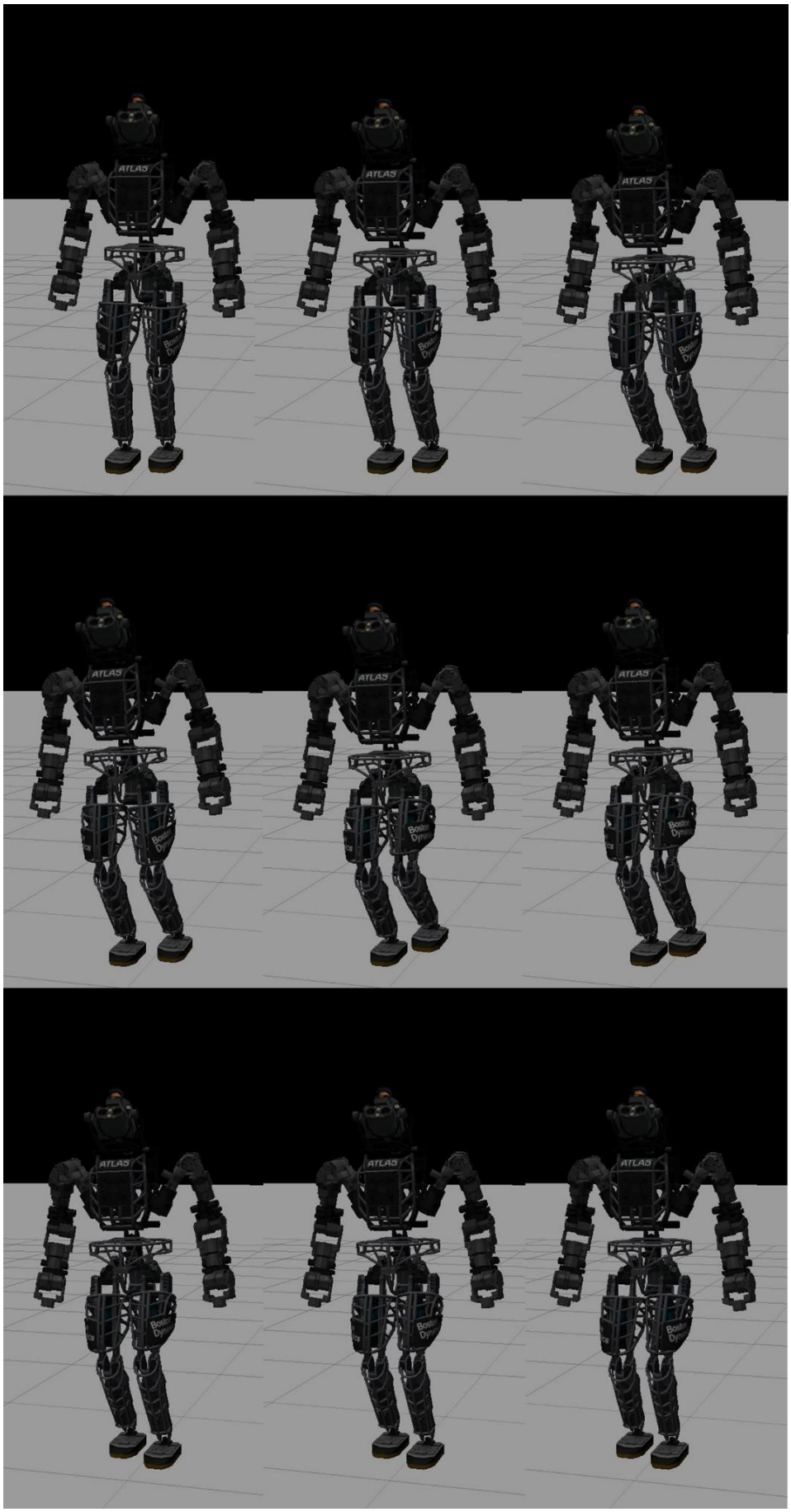

Figura 36 - Primeira parte do ciclo de locomoção completo.

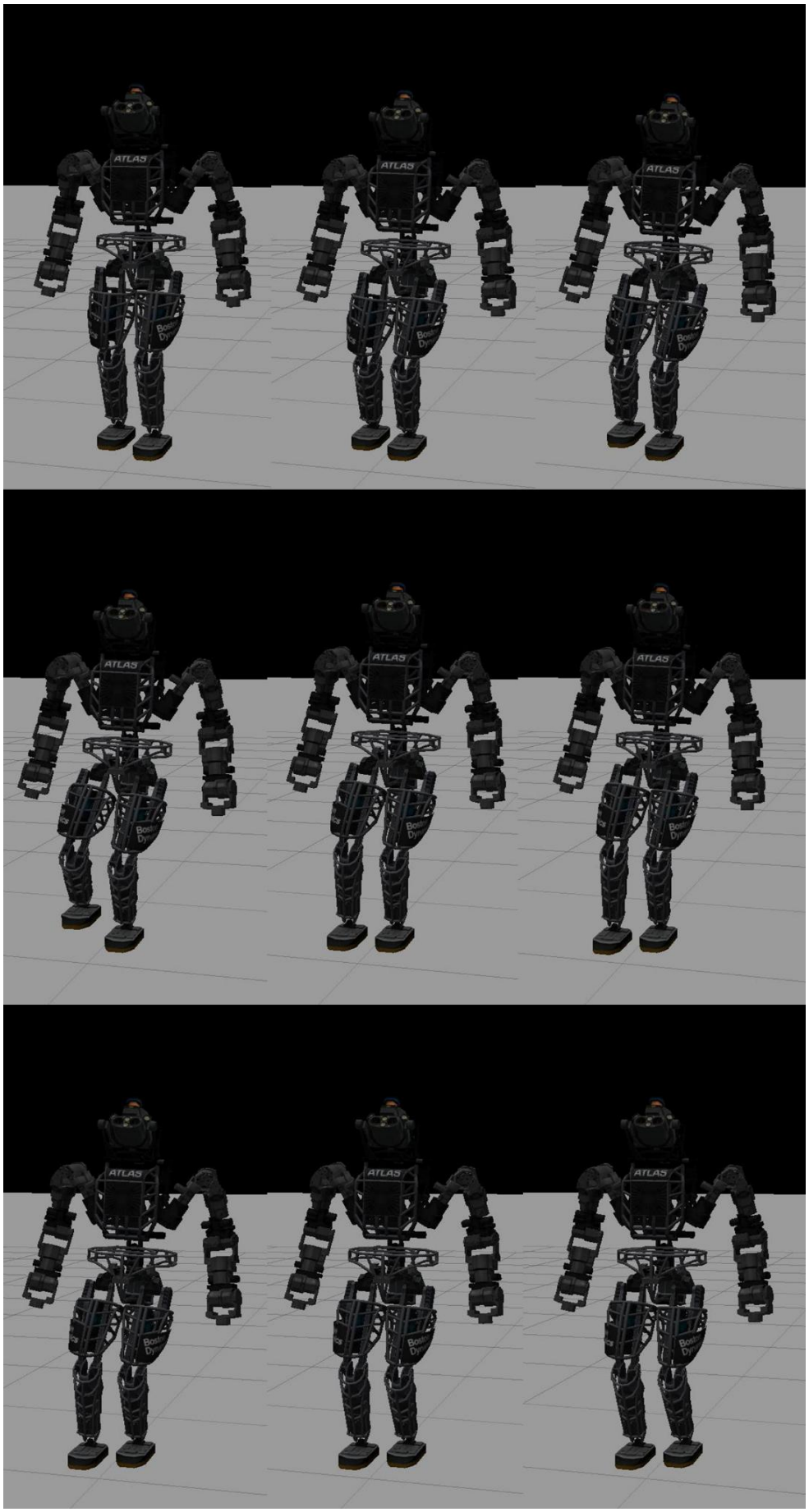

Figura 37 - Segunda parte do ciclo de locomoção completo.

Este capitulo apresentou os resultados obtidos utilizando-se o controle proposto. Avaliou-se também uma abordagem diferenciada de treinamento para o aprendizado por reforço com resultados satisfatórios. No capítulo seguinte serão apresentadas as conclusões e serão sugeridos possíveis trabalhos futuros.

6.
Conclusão

Esta dissertação apresentou um modelo de controle por torques computados de um robô humanoide Atlas, com um planejamento de locomoção via aprendizado por reforço pelo método *Q-Learning* aproximado. Os resultados dos testes realizados com tal modelo permitem concluir que o modelo hibrido possibilita a caminhada do robô em regime de locomoção estática, de acordo com o objetivo proposto neste trabalho. O controle demonstrou ser adaptável aos ruídos introduzidos pelos instrumentos simulados e foi capaz de redefinir sua atuação no ambiente do robô simulado, que era consideravelmente distinto do utilizado para o treinamento prévio dos agentes. Por conta das restrições introduzidas, não houve incidente de queda do modelo robótico, permitindo afirmar, assim, que o controle desenvolvido tem grandes probabilidades de êxito se aplicado diretamente a um robô real.

Como o controle foi projetado para uma locomoção retilínea em ambiente controlado, trabalhos futuros poderão tratar de um desenvolvimento do modelo para operação em ambientes de solo irregular e locomoção em outras direções. O modelo também pode ser adaptado para um regime de locomoção dinâmica, para permitir caminhadas mais rápidas e fluidas. Tal adaptação necessitaria de uma reformulação das variáveis que compõem o aprendizado por reforço, pois não necessariamente o centro de massa estaria em uma região pertencente ao polígono de suporte, como é o caso na implementação atual.

O modelo de controle desenvolvido pode ser aprimorado para, em um trabalho futuro, possibilitar a atuação simultânea das pernas e braços, resultando em locomoções ainda estáticas, mas mais fluidas. Pode-se explorar a utilização dos braços para manipulação de objetos – envolvendo a modelagem de um dos diversos módulos de mãos que podem ser fixadas aos pulsos do robô. Outro possível desenvolvimento seria utilizar os sistemas de visão já acoplados à cabeça do robô para mapear o ambiente ou determinar a forma de interação do robô com obstáculos e objetos.

7.
Referências bibliográficas

1 MASON, M. T. **Creation Myths: The Beginnings of Robotics Research**. Robotics & Automation Magazine, v.19, n.2, p.72-77, mai./jun. 2012.

2 INTERNATIONAL FEDERATION OF ROBOTICS. **History of Industrial Robots, from the first installation until now**. Desenvolvido por: International Federation of Robotics. Disponível em: <http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf>. Acesso em: 12 jan. 2016.

3 VUKOBRATOVIC, M.; JURICIC, D.; FRANK, A. A. **On the Control and Stability of One Class of Biped Locomotion Systems**. Journal of Basic Engineering, v.92, n.2, p.328-332, jun. 1970.

4 JURICIC, D.; VUKOBRATOVIC, M. **Mathematical Modelling of a Bipedal Walking System**, ASME WINTER ANNUAL MEETING, 9. 1972, New York. p.1-8, 26-30 nov. 1972.

5 VUKOBRATOVIC, M.; STOKIC, D. **Dynamic Stability of Unstable Legged Locomotion Systems**. Mathematical Biosciences, v.24, n.1-2, p.129-157, Jan./jun. 1975.

6 TEDRAKE, R. 6.832 **Underactuated Robotics**. Spring, 2009. Massachusetts Institute of Technology: MIT OpenCourseWare. Disponível em: <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-832-underactuated-robotics-spring-2009/readings/MIT6_832s09_read_ch01.pdf> Acesso em: 12 jan. 2016.

7 XUAN, X. et al. **Limit cycle walking of underactuated bipedal humanoid on slippery road surface**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 14. 2014, Madrid. p.622-627, 18-20 nov. 2014.

8 MASON, S.; RIGHETTI, L.; SCHAAL, S. **Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 14. 2014, Madrid. p.374-379, 18-20 nov. 2014.

9 HERZOG, A. et al. **Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics**. INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 27. 2014, Chicago. p.981-988, 14-18 set. 2014.

10 HULEA, M.; CARUNTU, C. F. Spiking **neural network for controlling the artificial muscles of a humanoid robotic arm**. INTERNATIONAL CONFERENCE ON SYSTEM THEORY, CONTROL AND COMPUTING, 18. 2014, Sinaia. p.163-168, 17-19 out. 2014.

11 HULUTA, E.; DA SILVA, R. F.; OLIVEIRA, T. E. A. **Neural network-based hand posture control of a humanoid robot hand**. INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND VIRTUAL ENVIRONMENTS FOR MEASUREMENT SYSTEMS AND APPLICATIONS, 2. 2014, Ottawa. p.124-128, 5-7 mai. 2014.

12 ZHI, L. et al. **Adaptive neural control for dual-arm coordination of humanoid robot with unknown nonlinearities in output mechanism**. IEEE Transactions on Cybernetics, v.45, n.3, p.507-518, mar. 2015.

13 YANG, L.; LIU, Z.; ZHANG, Y. **Adaptive fuzzy yaw moment control of humanoid robot based on ankle joint**. INTERNATIONAL CONFERENCE ON CHINESE CONTROL CONFERENCE, 34. 2015, Hangzhou. p.5999-6004, 28-30 jul. 2015.

14 YUNDUAN, C.; MATSUBARA, T.; SUGIMOTO, K. **Local Update Dynamic Policy Programming in reinforcement learning of pneumatic artificial muscle-driven humanoid hand control**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 15. 2015, Seoul. p. 1083-1089, 3-5 nov. 2015.

15 HWANG, K.S.; LING, J.L.; WEI-HAN WANG. **Adaptive reinforcement learning in box-pushing robots**. INTERNATIONAL CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING, 10. 2014, Taipei. p.1182-1187, 18-22 ago. 2014.

16 HWANG, C. L.;LAN, C. W.; CHOU, Y. J. **Search, Track, and Kick to Virtual Target Point of Humanoid Robots by a Neural-Network-Based Active Embedded Vision System**. IEEE Systems Journal, v.9, n.1, p.107-118, mar. 2015.

17 ERCELIK, E.; SENGOR, N. S. **A neurocomputational model implemented on humanoid robot for learning action selection**. INTERNATIONAL JOINT CONFERENCE NEURAL NETWORKS, 25. 2015, Killarney. p.1-6, 12-17 jul. 2015.

18 FOLGHERAITER, M.; TAZHIGALIYEVA, N.; NIYETKALIYEV, **A. Adaptive joint trajectory generator based on a chaotic recurrent neural network**. INTERNATIONAL CONFERENCE ON DEVELOPMENT AND LEARNING AND EPIGENETIC ROBOTICS, 11. 2015, Providence. p.285-290, 13-16 ago. 2015.

19 SAPUTRA, A.A.; TAKEDA, T.; KUBOTA, N. **Efficiency energy on humanoid robot walking using evolutionary algorithm**. CONGRESS ON

EVOLUTIONARY COMPUTATION, 17. 2015, Sendai. p.573-578, 25-28 mai. 2015.

20 MAOUDJ, A.; BOUZOUIA, B.; HENTOUT, A.; TOUMI, R. **Multi-agent approach for task allocation and scheduling in cooperative heterogeneous multi-robot team: Simulation results**. INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, 13. 2015, Cambridge. p.179-184, 22-24 Jul. 2015.

21 LAMINI, C.; FATHI, Y.; BENHLIMA, S. **Collaborative Q-learning path planning for autonomous robots based on holonic multi-agent system**. INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS: THEORIES AND APPLICATIONS, 10. 2015, Rabat. p.1-6, 20-21 out. 2015.

22 OSKIN, B. **Japan Earthquake & Tsunami of 2011: Facts and Information**. Live Science, New York, 07 mar. 2015. Disponível em: <http://www.livescience.com/39110-japan-2011-earthquake-tsunami-facts.html>. Acesso em: 12 jan. 2016.

23 STRICKLAND, E. **Meet the Robots of Fukushima Daiichi**. IEEE Spectrum, New York, 28 fev. 2014. Disponível em: <http://spectrum.ieee.org/slideshow/robotics/industrial-robots/meet-the-robots-of-fukushima-daiichi>. Acesso em: 12 jan. 2016.

24 DARPA Robotics Challenge. **Overview**. Desenvolvido por: DARPA. Disponível em: <http://www.theroboticschallenge.org/overview>. Acesso em: 12 jan. 2016.

25 DARPA Robotics Challenge. **The Value of Open Source Simulation for Robotics Development and Testing**. Desenvolvido por: DARPA. Disponível em: <http://www.theroboticschallenge.org/blog/open_source_simulation>. Acesso em: 12 jan. 2016.

26 HORNYAK, T. Be **afraid: DARPA unveils Terminator-like Atlas robot**. CNet, California, 11 jul. 2013. Disponível em: <http://www.cnet.com/news/be-afraid-darpa-unveils-terminator-like-atlas-robot/>. Acesso em: 12 jan. 2016.

27 MARKOFF, J. **Modest Debut of Atlas May Foreshadow Age of 'Robo Sapiens'**. The New York Times, New York, 11 jul. 2013. Disponível em: <http://www.nytimes.com/2013/07/12/science/modest-debut-of-atlas-may-foreshadow-age-of-robo-sapiens.html?_r=0>. Acesso em: 12 jan. 2016.

28 LUND, H. H.; MIGLINO, O. **From simulated to real robots**. INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION, 3. 1996, Nagoya. p.362-365, 20-22 mai. 1996.

29 SHAFI, N.; REIS, L. P.; ROSSETTI, R. J. F. **Two humanoid simulators: Comparison and synthesis**. IBERIAN CONFERENCE ON

INFORMATION SYSTEMS AND TECHNOLOGIES, 6. 2011, Chaves. p.1-6, 15-18 jun. 2011.

30 GAZEBO. **Home**. Desenvolvido por: Open Source Robotics Foundation. Disponível em: <http://gazebosim.org/>. Acesso em: 12 jan. 2016.

31 ROS. **About ROS**. Desenvolvido por: Open Source Robotics Foundation. Disponível em: <http://www.ros.org/about-ros/>. Acesso em: 12 jan. 2016.

32 ASADA, H. H. **Introduction to Robotics**. Fall, 2005. Massachusetts Institute of Technology: MIT OpenCourseWare. Disponível em: <http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter3.pdf>. Acesso em: 12 jan. 2016.

33 HIBBELER. R. C. **Engineering Mechanics: Dynamics**. 13.ed. New York: Prentice Hall, 2012. 768p.

34 ASADA, H.; SLOTINE, J. J. **Robot Analysis and Control**, 1.ed. New York: Wiley, 1986. 288p.

35 DENAVIT, J.; HARTENBERG, R. S. **A kinematic notation for lower-pair mechanisms based on matrices**. Journal of Applied Mechanics, v.23, n.2, p.215-221, jun. 1955.

36 WATSON G. Right **Hand Rule for Cross Products**. Fall, 1998. University of Delaware. Disponível em: <http://www.physics.udel.edu/~watson/phys345/Fall1998/class/1-right-hand-rule.html>. Acesso em: 12 jan. 2016.

37 MISTRY, M.; SCHAAL, S.; YAMANE, K. **Inertial parameter estimation of floating base humanoid systems using partial force sensing**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 9. 2009, Paris, p.492-497, 7-10 dez. 2009.

38 BOUYARMANE, K.; KHEDDAR, A. **On the dynamics modeling of free-floating-base articulated mechanisms and applications to humanoid whole-body dynamics and control.** INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 12. 2012, Osaka. p.36-42, 29 nov.-01 dez. 2012.

39 SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot Modeling and Control**. 1.ed. New York: Wiley, 2006. 496p.

40 ASADA, H. H. **Introduction to Robotics**. Fall, 2005. Massachusetts Institute of Technology: MIT OpenCourseWare. Disponível em: <http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter7.pdf>. Acesso em: 13 jan. 2016.

41 BURSCHKA, D. **Robotik**. Winter, 2006. Technische Universität München. Disponível em: <http://www6.in.tum.de/burschka/courses/robotics/aufgaben/solution04.pdf>. Acesso em: 13 jan. 2016.

42 LEWIS, F. L.; DAWSON, D. M.; ABDALLAH, C. T. **Robot Manipulator Control: Theory and Practice**. 2.ed. New York: Marcel Dekker inc., 2004. 638p.

43 GOSWAMI, D. **Biped locomotion: stability analysis, gait generation and control.** Singapora, 2009. 197p. Tese (Doutorado em Engenharia) – Departamento de Engenharia Elétrica e Computação, National University of Singapore.

44 ZHANG, B.; CHEN S.; XU M. **Entry trajectory generation based on neural network**. INTERNATIONAL CONFERENCE ON ELECTRONIC AND MECHANICAL ENGINEERING AND INFORMATION TECHNOLOGY, 1. 2011, Harbin. p.2998-3001, 12-14 ago. 2011.

45 KAIDI, R.; LAZAAR, M.; ETTAOUIL, M. **Neural network apply to predict aircraft trajectory for conflict resolution**. INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS: THEORIES AND APPLICATIONS, 9. 2014, Rabat, p.1-6, 7-8 mai. 2014.

46 FOLGHERAITER, M.; TAZHIGALIYEVA, N.; NIYETKALIYEV, A. **Adaptive joint trajectory generator based on a chaotic recurrent neural network**. INTERNATIONAL CONFERENCE ON DEVELOPMENT AND LEARNING AND EPIGENETIC ROBOTICS, 11. 2015, Providence. p.285-290, 13-16 ago. 2015.

47 CHEN, Y.; TSENG, C. **A robust fuzzy trajectory estimation design of high speed reentry vehicles**. INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS, 20. 2011, Taipei. p.368-373, 27-30 jun. 2011.

48 TUDOR, L.; MOISE, A. **Automatic expert system for fuzzy control of robot trajectory in joint space**. INTERNATIONAL CONFERENCE ON MECHATRONICS AND AUTOMATION, 10. 2013, Takamatsu. p.1057-1062, 4-7 ago. 2013.

49 KIM, S. et al. **Trajectory planning of autonomous robot using advanced fuzzy controller**. INTERNATIONAL CONFERENCE ON INFORMATION AND AUTOMATION, 4. 2010, Harbin. p.482-485, 20-23 jun. 2010.

50 SUGIMOTO, N.; MORIMOTO, J. **Phase-dependent trajectory optimization for CPG-based biped walking using path integral reinforcement learning**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 11. 2011, Bled. p.255-260, 26-28 out. 2011.

51 SUGIMOTO, N.; MORIMOTO, J. **Trajectory-model-based reinforcement learning: Application to bimanual humanoid motor learning with a closed-chain constraint**. INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS, 13. 2013, Atlanta. p.429-434, 2013. 15-17 out. 2013.

52 GREENWOOD, G. W. et al. **Online generation of trajectories for autonomous vehicles using a multi-agent system**. CONGRESS ON

EVOLUTIONARY COMPUTATION, 16. 2014, Beijing. p.1218-1224, 6-11 jul. 2014.

53 ENCYCLOPAEDIA BRITANNICA. **Machine Learning, Artificial Intelligence**. Desenvolvido por: Encyclopaedia Britannica, inc. Disponível em: <http://global.britannica.com/technology/machine-learning>. Acesso em: 13 jan. 2016.

54 NILSSON, N. J. **Introduction to Machine Learning. Robotics Laboratory**. Department of Computer Science. 2.ed. California: Stanford University. 1998. 188p.

55 MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. 1.ed. Cambridge: The MIT Press, 2012. 432p.

56 SUTTON R.S.; BARTO A. G. **Reinforcement Learning: An Introduction**. 1.ed. Cambridge: The MIT Press, 1998. 322p

57 BELLMAN, R. **A Markovian Decision Process**. Journal of Mathematics and Mechanics, v.6, n.4, p.679-684, dez. 1957.

58 FEINBERG E.A. (Ed.); SHWARTZ A. (Ed.). **Handbook of Markov Decision Processes**. 1.ed. Boston: Kluwer. 2002. 565p

59 RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3.ed. New York: Prentice Hall. 2010. 946p

60 BELLMAN, R. **On the Theory of Dynamic Programming**. Proceedings of the National Academy of Sciences, v.38, n.8, p.716-719, ago. 1952.

61 SUTTON, R. S. **Learning to predict by the methods of temporal difference.** Machine Learning. Kluwer Academic Publisher, v.3, n.1, p.9-44. ago. 1988.

62 ZHENG, Y.; LUO, S.; ZHANG, J. **Greedy exploration policy of Q-learning based on state balance.** CONFERENCE ON CONVERGENT TECHNOLOGIES FOR THE ASIA-PACIFIC, 17. 2005, Melbourne. p.1-4, 21-24 nov. 2005.

63 GAO, Q. et al. **An Improved Q-learning Algorithm Based on Exploration Region Expansion Strategy.** WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 6. 2006, Dalian. p.4167-4170. 21-23 jun. 2006.

64 SHEN, Y.; ZENG, C. **An Adaptive Approach for the Exploration-Exploitation Dilemma in Non-stationary Environment.** INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING, 1. 2008, Wuhan. p.497-500, 12-14 dez. 2008.

65 PANDEY, D.; PANDEY, P. **Approximate Q-Learning: An Introduction.** INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND COMPUTING, 2. 2010, Bangalore. p.317-320, 09-11 fev. 2010.