Relatório Final de Atividades

Controle de Estabilização de Caminhada de Robô Humanoide

Bolsa de Iniciação Científica 2020/04559-6
Período relatado: 10/12/2020 a 30/06/2021
Período de vigência: 01/07/2020 a 30/06/2021
Instituto Tecnológico de Aeronáutica – ITA

Aluno: Reynaldo Santos de Lima
Orientador: Marcos Ricardo Omena de Albuquerque Maximo

9 de julho de 2021

Resumo

Neste trabalho estudam-se adaptações em algoritmos de estabilização de caminhada em robô humanoide de baixo custo (ITAndroids Chape 1ª e 2ª gerações). Esse trabalho será realizado no Laboratório de Sistemas Computacionais Autônomos (LAB-SCA), onde desenvolvem-se robôs humanoides. São apresentados aspectos importantes do modelo em baixo nível do controle, com a equação do manipulador ilustrando a dinâmica utilizada para cada plano considerado no controle de estabilização (coronal e sagital). Dá-se destaque à interface prática das diferentes ferramentas utilizadas no desenvolvimento e projeto dos ganhos do controlador, bem como as melhorias adotadas neste trabalho com o objetivo de tornar o código humanoide mais seguro, mais modularizado e de manutenibilidade maior.
Palavras chaves: Caminhada de robôs humanoides, Controle, Robótica.

Sumário

1 Resumo do plano inicial

O plano inicial, do trabalho de 12 meses, com início em julho de 2020, dividido em execução nas atividades listadas a seguir:

A Estudo introdutório da matéria de controle;

B Estudo sobre algoritmos de otimização;

C Estudo sobre a caminhada do robô humanoide;

D Estudo detalhado do código do robô humanoide relacionado ao seu caminhar, organização do código e início dos planos de otimização;

E Confecção do primeiro relatório científico;

F Continuação do processo de otimização;

G Testes no robô real seguidos de aquisição de dados;

H Fim do processo de otimização seguido da comparação de resultados do antes e pós processo de modificação e otimização;

I Implementação definitiva do novo código;

J Ajuste na malha de controle para melhor desempenho;

K Confecção do segundo relatório científico.

Tabela 1: Cronograma de atividades detalhado.

Bimestre	Atividade										
	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6											

2 Resumo das etapas realizadas

Foram realizadas as seguintes atividades no período de 11/06/2021 a 30/06/2021:

A Continuação do processo de otimização;

B Reestruturação do mapeamento de parâmetros no código;

C Ajuste de ganhos para controladores do plano sagital;

D Ajuste de ganhos para controladores do plano coronal;

E Implementação definitiva do novo código;

F Confecção do segundo relatório científico.

3 Código de estabilização de caminhada

Na implementação do controlador de caminhada humanoide, a malha de controle é dividida em vários níveis, dada a complexidade do problema. Utiliza-se, comumente, o modelo do pêndulo invertido com o centro de massa do robô [1]. Já em uma camada mais baixo nível do controle, está o compensador de orientação do torso, o qual garante a estabilização da caminhada junto ao controle de compensação de gravidade.

Na Figura 1, tem-se a malha de controle completa esquematizada para a caminhada, destacando-se o compensador de orientação de torso, o qual foi o foco das melhorias estudadas. Para a construção do controle nesta malha (compensador de orientação de torso), utilizam-se controladores P+V desacoplados para os canais de arfagem e rolagem. Ademais, o robô humanoide estudado é controlado por posição, sendo as orientações estimadas utilizando-se um filtro de Kalman estendido (EKF) que usa as informações do acelerômetro e do girômetro da unidade inercial presente no torso do robô, estratégia implementada em [2].

Escolhe-se um controlador P+V e não um PD de modo a evitar amplificação de ruído por meio da derivação das estimativas de ângulo. Utilizam-se, então, as medidas de

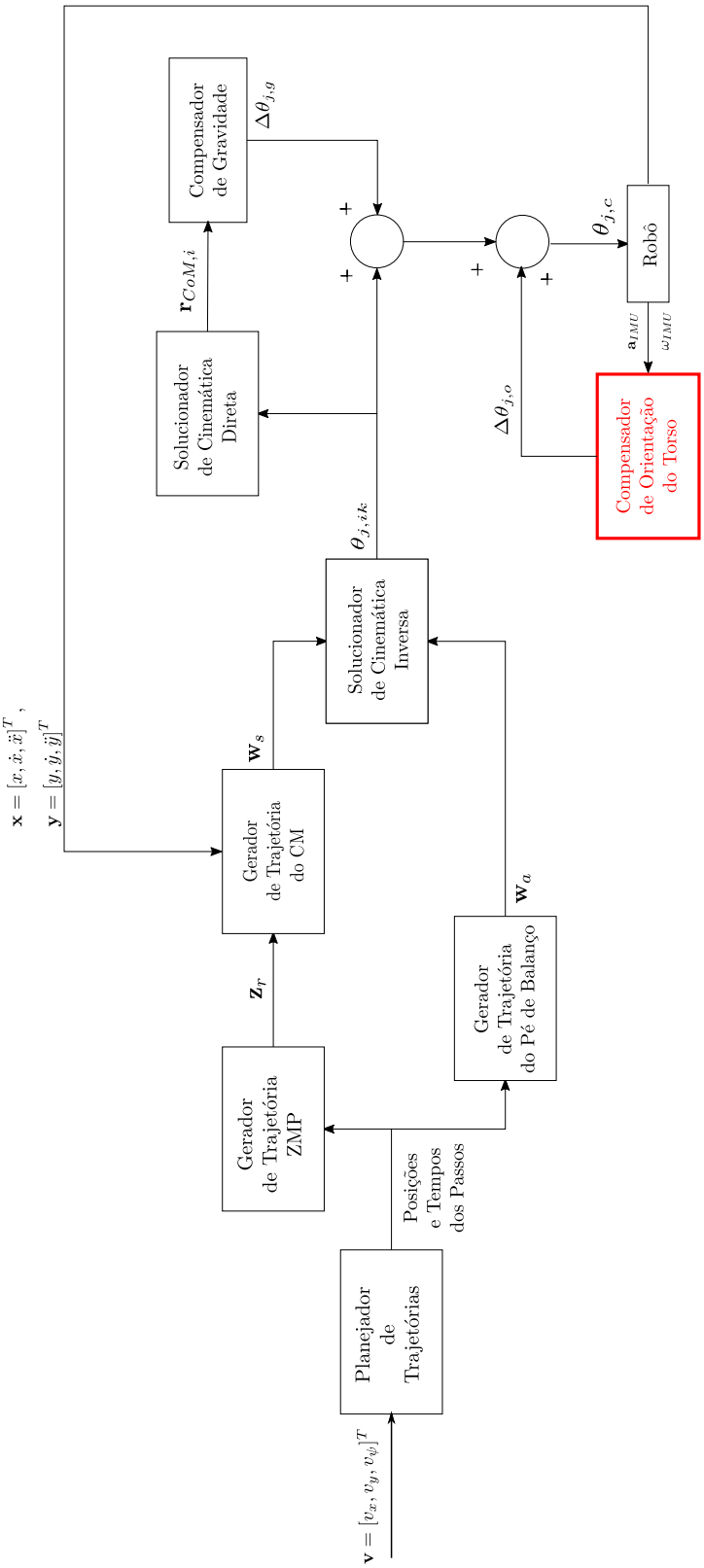


Figura 1: Visão geral do controle de caminhada.

velocidade angular do girômetro. O controlador comanda o deslocamento de posição como:

$$\Delta\theta_{hr,o} = -\left[K_{p,hr}\left(\phi_d - \phi\right) - K_{v,hr}p\right], \tag{1}$$

$$\Delta\theta_{ar,o} = -\left[K_{p,ar}\left(\phi_d - \phi\right) - K_{v,ar}p\right], \tag{2}$$

$$\Delta\theta_{hp,o} = -\left[K_{p,hp}\left(\theta_d - \theta\right) - K_{v,hp}q\right], \tag{3}$$

$$\Delta\theta_{kp,o} = -\left[K_{p,kp}\left(\theta_d - \theta\right) - K_{v,kp}q\right], \tag{4}$$

$$\Delta\theta_{ap,o} = -\left[K_{p,ap}\left(\theta_d - \theta\right) - K_{v,ap}q\right], \tag{5}$$

em que ϕ_d e θ_d são ângulos desejados de rolamento e arfagem, respectivamente; ϕ e θ são ângulos atuais de rolamento e arfagem, respectivamente; p e q são velocidades angulares de rolamento e arfagem, respectivamente; os índices hr, ar, hp, kp e ap referem-se a coxa-rolamento (*hip-roll*, em inglês), calcanhar-rolamento (*ankle-roll*, em inglês), coxa-arfagem (*hip-pitch*, em inglês), joelho-arfagem (*knee-pitch*, em inglês) e calcanhar-arfagem (*ankle-pitch*, em inglês), respectivamente; $K_{p,*}$ e $K_{v,*}$ representam ganhos proporcionais e derivativos de cada junta, respectivamente; finalmente, $\Delta\theta_{*,o}$ representa o comando de deslocamento (*offset*) de cada junta.

Vale ressaltar que, para determinar os ganhos, lineariza-se o modelo não linear das equações do manipulador em torno de uma postura nominal (robô com velocidade zero no meio da fase de duplo suporte) [2]. A linearização é realizada pelo método de pequenos sinais [3].

Esta modelagem é especialmente útil, visto que o robô humanoide pode ser considerado um manipulador completamente atuado se seu *zero moment point* (ZMP) estiver dentro do polígono de suporte. Neste caso, considera-se que o pé do robô está preso no solo [4]. O ZMP é um conceito popular na literatura de caminhada, sendo o ponto no qual pode-se considerar que as forças de reação devem atuar para equilibrar o mecanismo bípede [4]. Quando o ZMP está dentro do polígono de suporte, este coincide com o centro de pressão da base.

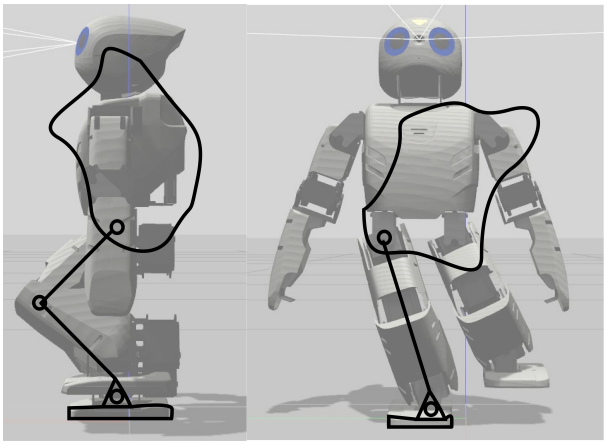


Figura 2: Simplificação do robô para manipuladores no plano sagital (esquerda) e coronal (direita).

Nota-se que os critérios de estabilidade utilizados mais corriqueiramente, como o critério do ZMP dentro do polígono de suporte, tratam da estabilidade local da caminhada, isto é, observa-se as condições de estabilidade para o estado atual do robô a cada passo, não para a caminhada como um todo. O fato da análise de estabilidade ser tratada localmente tornam os critérios estabelecidos restritivos, produzindo um movimento de caminhada muito aquém do melhor movimento quanto ao aproveitamento energético. Para comparação, estima-se que o robô ASIMO utiliza 10 vezes mais energia que um ser humano para caminhar [5].

Como exposto, consideram-se dois manipuladores robóticos desacoplados, um para o plano sagital (arfagem) e outro para o plano coronal (rolamento). A Figura 2 mostra como decorre-se a estruturação dos dois manipuladores, em que tem-se como bases os pés de suporte. É projetado um controlador SISO para cada junta de modo a respeitar restrições de interesse (banda passante e margem de fase) com auxílio de um algoritmo de otimização Nelder-Mead (da função fminsearch do MATLAB).

No desenvolvimento implementado pelo LAB-SCA em seu projeto de robôs humanoide, a interface da obtenção de parâmetros até a implementação dos ganhos $K_{p,*}$ e $K_{v,*}$ projetados, tem-se a interface de diferentes ferramentas. Obtém-se parâmetros a partir do modelo em CAD do robô e de medições do próprio robô físico. O modelo de dinâmica do robô e a malha de controle final completa é implementada em linguagem

C++. Como mencionado, a determinação dos ganhos é feita em MATLAB. Na análise entre essas diferentes camadas de ajuste da malha de controle relacionada à compensação de orientação de torso, observaram-se algumas melhorias a serem realizadas no código da equipe.

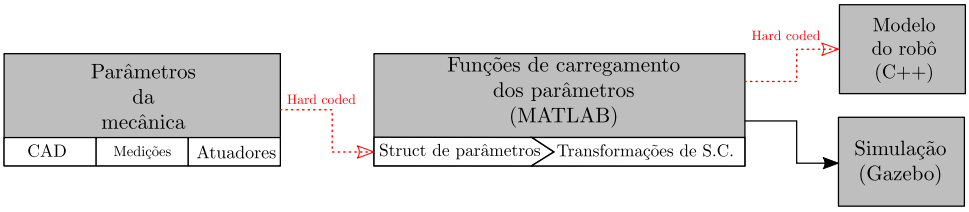
3.1 Reestruturação na obtenção de parâmetros

A primeira etapa para a construção do modelo do robô humanoide é o correto ajuste de parâmetros fundamentais do robô. Para isso, utilizam-se parâmetros disponíveis a partir da montagem em CAD da mecânica do robô, como a massa, o centro de massa e o tensor de inércia de cada subsistema do robô. Outros parâmetros, como a massa total do robô, são retirados diretamente pela medição do robô físico ou de parâmetros do modelo dos atuadores. Para cada modelo de robô constrói-se uma struct dos parâmetros.

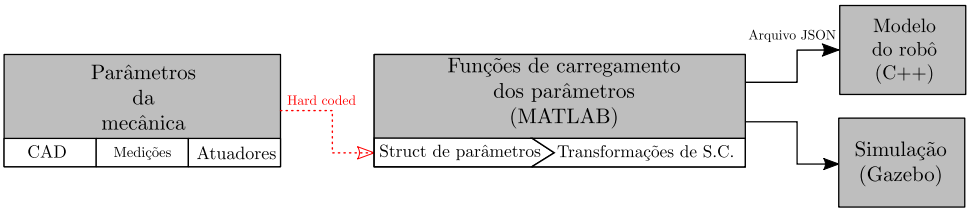
Com a struct dos parâmetros para cada robô, adequa-se os parâmetros num sistema de coordenadas (S.C.) novo, com relação a uma origem. Esta origem pode ser o próprio centro de massa ou outro ponto de interesse fixo, sendo este último mais vantajoso pois o centro de massa muda constantemente no espaço de acordo com o movimento do robô. O método a realizar estas transformações é feito pela equipe no MATLAB, pois extrai informações uteis tanto para o código em C++ como para o modelo de simulação em Gazebo [6].

Durante o estudo do código para otimização, porém, notou-se que a forma que esta etapa foi implementada era de baixa manutenibilidade. Extraía-se as informações da estrutura de parâmetros do MATLAB e estas eram colocadas no código C++ de forma *hard coded*, ou seja, repetia-se a escrita dos parâmetros do robô, agora transformadas para um novo sistema de coordenadas, no código em C++ do modelo do robô.

De modo a corrigir este problema, implementou-se a interface automática entre a estrutura de parâmetros construída no MATLAB e o modelo do robô no código principal em C++ por meio de uma árvore de parâmetros em arquivo JSON. Assim, os parâmetros gerais do robô são passados apenas uma vez a um script no MATLAB e ao carregar os



(a) Sem interface de arquivo JSON.



(b) Com interface por JSON implementada.

Figura 3: Interface do carregamento e transformação dos parâmetros do robô.

parâmetros para a simulação em gazebo, também constrói-se um arquivo JSON que é lido pelo código principal em C++.

Com isto, a estrutura do código como um todo ficou mais modularizada e menos propícia a erros na escrita dos parâmetros. No MATLAB utilizou-se a função *jsonencode* e, no C++, a biblioteca *Boost* [7]. Um diagrama esquematizado do antes e depois da estrutura de obtenção dos parâmetros é apresentada na Figura 3, em que destaca-se a utilização do arquivo JSON para evitar uma trilha de código *hard coded*.

3.2 Plano sagital: projeto de ganhos dos controladores

A Figura 4 mostra os ângulos base para os ângulos de arfagem do plano sagital $\theta_{*p}^{(0)}$, em que $*$ podem ser os subscritos a, k e h, que representam, respectivamente, a junta do tornozelo, junta do joelho e h, junta do quadril [8].

A interface para o projeto de ganhos era iniciada a partir do modelo do robô construído no código C++. Escolhe-se a construção do modelo desta forma, pois nesta etapa facilitam-se as operações de transformação para posições de interesse e permitem a construção de modelos particulares para cada unidade de robô com suas particularidades, não apenas iterando por modelo ou geração de robô.

Parte-se do modelo do robô em posição de T, isto é, braços completamente aber-

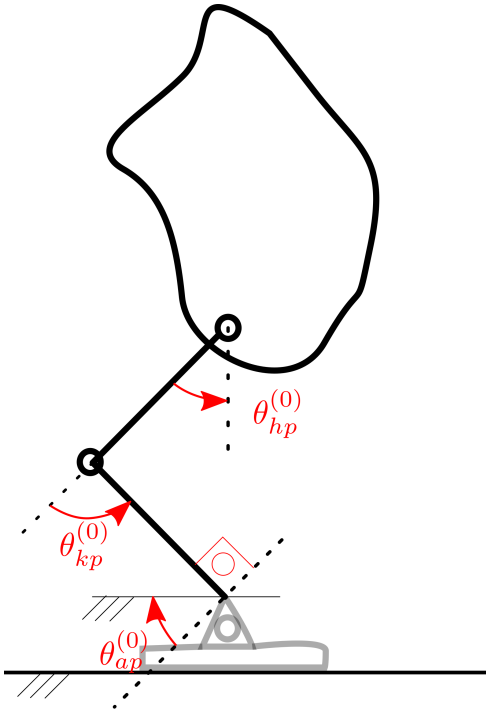


Figura 4: Representação em manipulador do plano sagital do robô humanoide.

tos, pernas retas e estendidas. Utilizando as bibliotecas desenvolvidas em C++ pelo laboratório LAB-SCA, são feitas as transformações do modelo para obter-se uma representação da posição assumida como padrão para o controle.

Com esta posição, calculam-se o modelo de manipulador do plano sagital (pêndulo invertido de três elos), supondo que:

- Uma das pernas é assumida como pé de suporte no movimento de caminhada;
- Perna e coxa constituem os dois primeiros elos do manipulador;
- O restante do corpo do robô e o pé flutuante constituem o terceiro elo do manipulador.

Com este modelo, obtém-se do código C++ os vetores, massas e inércias dos elos descritos, formando a cadeia cinemática para o modelo.

Com isso, é possível construir, no ambiente do MATLAB, o modelo dinâmico, apli-

cando a equação do manipulador:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathfrak{T}, \tag{6}$$

em que $\mathbf{q}$ é o vetor de estado dos ângulos no plano sagital, $\mathfrak{T}$ é o torque de entrada no sistema e as matrizes $\mathbf{B}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ e $\mathbf{g}(\mathbf{q})$ representam, respectivamente, a matriz de inércia, efeito coriolis e gravidade da equação do manipulador [9], obtidas com auxílio dos valores para a cadeia cinemática.

Obtendo o modelo dinâmico, lineariza-se o sistema como já discutido, sendo o cálculo das matrizes jacobianas do sistema feito com o auxílio do pacote de matemática simbólica do MATLAB. Ademais, com as restrições para os servos bem conhecidas, constrói-se uma função custo e obtém-se os ganhos da otimização (ver Seção 4 do relatório parcial [10]). Os ganhos da malha P+V do plano sagital eram, então, colocados num arquivo de configuração e serviam de entrada para o código do controle do robô, em C++.

O modelo implementado, porém, continha problemas na interface construída, algumas das quais foram corrigidas, como:

P.1. O código de Projeto de controladores sagital era feito num *(*script) que precisava de variáveis no *workspace* de dois scripts auxiliares referentes à construção do modelo dinâmico, os quais precisavam ser chamados manualmente em uma ordem específica.
Solução: Isolamento dos scripts auxiliares em funções, com o script de Projeto de controladores chamando as funções.

P.2. O conjunto das funções era feito sem levar em consideração diferentes modelos e gerações de robôs, de modo que os ganhos projetados não levavam em consideração as individualidades do robô.
Solução: Ajuste para funcionamento do código com entrada do usuário para qual geração do humanoide pretendia-se o cálculo dos ganhos.

Foi notado mais uma melhoria a ser aplicada, que ainda não foi feita:

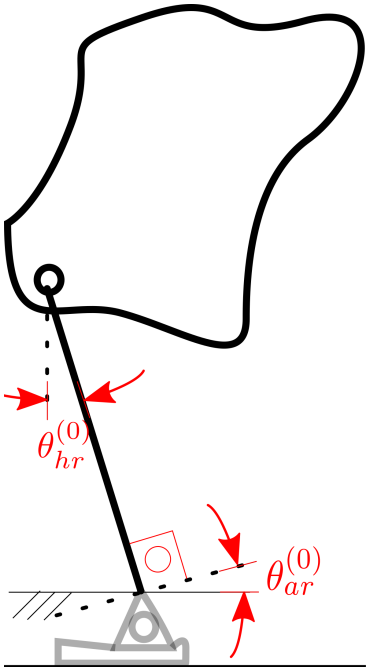


Figura 5: Representação em manipulador do plano coronal do robô humanoide.

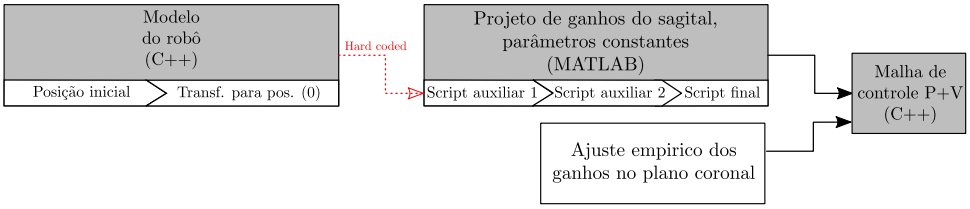
P.3. A passagem de cadeia cinemática do C++ é feita *hard coded*, ou seja, escreve-se diretamente no script MATLAB os valores obtidos em C++ para os elos do manipulador.

Solução a ser implementada: fazer o código de projeto de controladores sagital rodar o binário do C++ que constrói a cadeia cinemática e obter os parâmetros para a versão mais recente do projeto.

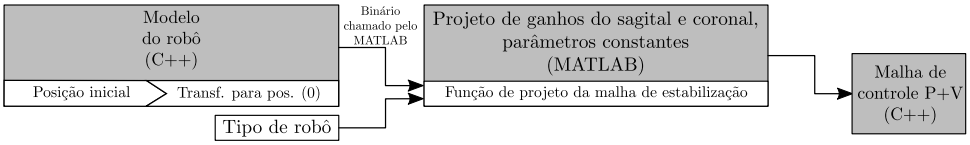
3.3 Plano coronal: projeto de ganhos dos controladores

A Figura 5 mostra os ângulos base para os ângulos de rolamento do plano coronal. Destaca-se que para o manipulador coronal, no laboratório LAB-SCA, utilizavam-se ganhos $K_{p,ar}$, $K_{v,ar}$, $K_{p,hr}$ e $K_{v,hr}$ ajustados empiricamente. Desse modo, construiu-se um programa análogo ao mencionado na Subseção 3.2, já implementando as melhorias aos problemas P.1 e P.2.

Um diagrama expondo a interface para projeto dos ganhos de controladores como eram feitos anteriormente e como devem vir a ser implementados é mostrado na Figura



(a) Interface inicial.



(b) Interface proposta com melhorias.

Figura 6: Interface do projeto dos ganhos dos controladores.

6. Destaca-se que a solução ao trecho *hard coded* ainda deve ser implementada. Ajustes propostos como um todo permitem maior modularidade e diminui-se consideravelmente a chance de erros no projeto dos ganhos. Considerando, por exemplo, possíveis modificações da mecânica do robô, no modo implementado inicialmente, da Figura 6.a), deveria-se executar o modelo do robô em C++ e reescrever no MATLAB a nova cadeia cinemática, enquanto com as melhorias propostas, da Figura 6.b), essas alterações seriam feitas em uma camada inferior (diagrama da Figura 3).

4 Conclusão e trabalhos futuros

O trabalho desenvolvido foi proveitoso em identificar os principais pontos a serem otimizados na interface do código de robô humanoide implementado pelo laboratório LAB-SCA. Destacam-se as melhorias já aplicadas e os modelos de interação entre plataformas diferentes com o mínimo de uso de códigos *hard coded* (ver Figuras 3 e 6).

Destaca-se, ainda, do plano inicial da iniciação científica (ver Seção 1), que o item G teve sua atuação impactada diretamente pela pandemia do novo coronavírus, com a impossibilidade de acesso à estrutura física do LAB-SCA e, por conseguinte, dos robôs. O item H do planejamento inicial, por sua vez, foi parcialmente afetado, visto que a comparação do desempenho da obtenção dos ganhos não foi testada.

Vale ressaltar, por fim, que se esperam melhorias limitadas com o ajuste dos ganhos, visto que a implementação descrita, como um todo, é robusta e consideravelmente conservadora. Isso se deve aos critérios exigidos na modelagem da caminhada de humanoide, os quais são altamente restritivos quanto à estabilidade, como mencionado na Seção 3.

Para trabalhos futuros, planejam-se aperfeiçoar as melhorias mencionadas na estrutura do código, bem como o estudo de ferramentas de simulação que considerem o uso do modelo de manipulador completo nos termos de primeira ordem na redução dos motores, como discutido no desenvolvimento do modelo do manipulador no relatório parcial da iniciação científica [10].

Com os modelos estudados e aperfeiçoados, além de pontos levantados como pouco mencionados na literatura de robôs humanoide, pretende-se publicar resultados deste trabalho nos seguintes congressos:

- Congresso brasileiro de automática, CBA;
- Latin America Robotics Symposium, LARS;
- International congress of mechanical engineering, COBEM.

Vale ressaltar, ainda, que o tema desenvolvido na iniciação científica serviu de proposta para a entrada do beneficiário no programa de mestrado na graduação (PMG) do Instituto Tecnológico de Aeronáutica, no programa de Pós-Graduação em Eng. Eletrônica e Computação (EEC).

No mestrado, pretende-se desenvolver um controlador preditivo de posição do movimento de caminhada, assumindo as hipóteses de posição do ZMP como restrições ao controlador. Além disso, para a implementação deste controlador, pretende-se partir da construção de um estimador de estados, incluindo a posição, orientação, velocidade linear e velocidade angular a partir do conjunto de dados da unidade inercial e posição dos membros.

Referências

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi e H. Hirukawa, "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation", em *Proc. Int. Conf. Intelligent Robots and Systems*, Maui, EUA, 2001, pp. 239 – 246.

[2] M. R. O. de Albuquerque Maximo, "Automatic walking step duration through model predictive control", Tese de Doutorado, Instituto Tecnológico de Aeronáutica, 2017.

[3] G. F. Franklin, J. D. Powell e A. Emami-Naeini, *Sistemas de Controle para Engenharia*, 6th. Bookman, 2013, ISBN: 978-85-8260-067-2.

[4] M. Vukobratović e B. Borovac, "Zero-Moment Point – Thirty Five Years of Its Life", *Int. J. Humanoid Robots*, v. 1, n. 1, pp. 157 –173, 2004.

[5] S. H. Collins, A. Ruina, R. Tedrake e M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers", *Science*, v. 307, n. 5712, pp. 1082 –1085, 2005.

[6] N. Koenig e A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator", em *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[7] B. Software, *Boost C++ Libraries*, https://www.boost.org/, 2011.

[8] M. R. O. de Albuquerque Maximo, "Omnidirectional ZMP-Based Walking for a Humanoid Robot", Dissertation of Master of Science, Instituto Tecnológico de Aeronáutica, 2015.

[9] B. Siciliano, L. Sciavicco, L. Villani e G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010, ISBN: 1849966346.

[10] R. S. de Lima e M. R. O. de Albuquerque Maximo, “Relatório Parcial de Atividades: Controle de Estabilização de Caminhada de Robô Humanoide, Bolsa de Iniciação Científica 2020/04559-6”, Laboratório de Sistemas Computacionais Autônomos, Instituto Tecnológico de Aeronáutica, rel. técn., dez. de 2020.