Simulação de Sistemas Robóticos com ROS
Robotic Systems Simulation with ROS

João Vitor Garcia Carvalho[1], Mauricio Eiji Nakai[2]

RESUMO

Dentro do ambiente da robótica, o desenvolvimento de máquinas é composta de várias etapas, como design das peças ou modelo de controle, podendo ser necessária a remodelagem de algum determinado aspecto gerando um custo extra ao desenvolvimento. Uma forma de poder verificar um fator para um robô para que não haja a demanda da refacção é a simulação de um sistema robótico em um ambiente virtual. Este projeto propõe uma simulação com a utilização de ROS, um conjunto de ferramentas e bibliotecas que auxiliam no desenvolvimento de robô. A preparação da simulação passa por etapas como: modelagem do robô em URDF, seleção dos nós que farão a simulação e seleção do algoritmo de controle. Os resultados deste projeto são promissores comprovando a possibilidade da simulação com a utilização do Robot Operating System.
PALAVRAS-CHAVE: Robótica, ROS, Simulação

ABSTRACT

Within the robotics environment, the development of machines is made up of several stages, such as part design or control model, and it may be necessary to remodel a certain aspect, generating an extra cost for development. One way to verify a factor for a robot so that there is no need for refactoring is to simulate a robotic system in a virtual environment. This project proposes a simulation using the ROS, a set of tools and libraries that assist in robot development. The simulation preparation goes through steps such as: modeling the robot in URDF, selecting the nodes that will carry out the simulation and selecting the control algorithm. The results of this project are promising, proving the possibility of simulation using the Robot Operating System.
KEYWORDS: Robotic, ROS, Simulation

INTRODUÇÃO

O desenvolvimento de robôs passa por várias etapas, uma delas sendo a etapa de teste, a qual é necessária ter o robô montado para que se possa obter resultados. No entanto, para isso é obrigatório ter o robô físico e caso os resultados não sejam suficientes pode ser necessário o redesign de peças do robô. Uma das formas de evitar é realizar os teste em ambientes virtuais para obter resultados preliminares perante o robô. Este artigo trás um estudo de como esta simulação pode ser feita, juntamento com um teste com um robô bípede de 8 graus de liberdade.

Para poder simular sistemas complexos como o funcionamento de um robô é necessário uma gama de bibliotecas e ferramentas que possam ser usadas para representar cada componente mecânico e de programação na máquina. ROS - *Robot Operating System* satifaz essa necessidade trazendo um conjunto de bibliotecas e ferramentas que auxiliam na modelagem de robôs, desde seu design físico até a forma como ele reconhece o meio em que está (OPEN, 2023b).

[1] Bolsista do(a) CNPq. Universidade Tecnológica do Paraná, Apucarana, Paraná, Brasil. E-mail: jcarvalho.2020@alunos.utfpr.edu.br.
[2] Docente no Curso Engenharia de Computação. Universidade Tecnológica do Paraná, Apucarana, Paraná, Brasil. E-mail: mauricionakai@utfpr.edu.br. ID Lattes: 7376345732096655.

O princípio básico de funcinamento do ROS se dá pela presença de nós que podem enviar ou receber informações. Estas informações são publicadas em tópicos, os quais um nó pode estar inscrito para poder receber essa informação. No requisito de simulação, há uma ferramenta que simula o ambiente em que o robô estará e gera dados virtuais para que os robôs possam fazer a leitura do meio com sensores, esta ferramenta é chamada *Gazebo* (OPEN, 2023c).

Para poder fazer a simulação de um robô no Gazebo é necessário fazer o design do robô de acordo com o formato unificado de descrição de robôs, URDF - *Unified Robot Description Format*, fazer a programação do robô para saber qual movimento deve fazer e definir os nós que serão usados (MACIEL; HENRIQUES; LAGES, 2014).

DESIGN DO ROBÔ

Para poder representar uma máquina real em um ambiente virtual é necessário descrevê-la em URDF (OPEN, 2023a). Um arquivo URDF é escrito em xml onde cada tag define uma característica do robô. Este modelo tem dois principais componentes, os links e as juntas, sendo os links as peças físicas do robô, como parte do braço ou perna, e juntas a junção de dois links que podem fazer, ou não, algum movimento.

Um link é descrito pela tag "*link*"e seus elementos principais são: "*visual*", "*collision*"e "*inertial*". O primeiro elemento define o formato visual e a cor do robô, sem interferir nas características físicas dele (OPEN, 2023a). O segundo elemento define, da mesma forma que o elemento "visual", o formato do link, no entanto, agora com características físicas. Por fim, o terceiro elemento define as características inerciais do link, especificando sua massa e os valores de inércia. Um exemplo de link está descrito na Figura 1.

Figura 1 – Exemplo de Link em URDF

```
<link name="nome_link">
  <visual>
    <origin xyz = "0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size ="0 0 0" />
    </geometry>
  </visual>
  <collision>
    <origin xyz = "0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size ="0 0 0" />
    </geometry>
  </collision>
  <inertial>
    <origin xyz ="0 0 0" rpy="0 0 0"/>
    <mass value="1" />
    <inertia ixx="0.0" ixy ="0.0" ixz ="0.0" iyy="0.0" iyz="0.0" izz="0.0"/>
  </inertial>
</link>
```

Fonte: Autoria própria

A junta é descrita pela tag "*joint*", seus elementos principais são: "*parent*", "*child*", "*origin*"e "*axis*", sendo também necessário definir seu tipo na sua tag. O primeiro elemento contém o nome do link pai, o segundo do filho. O terceiro componente define a origem da junta, especificando um valor em cada eixo. O quarto componente define o eixo, ou os eixos, em que acontecerá o movimento. Um exemplo de junta está descrito na Figura 2.

Figura 2 – Exemplo de Junta em URDF

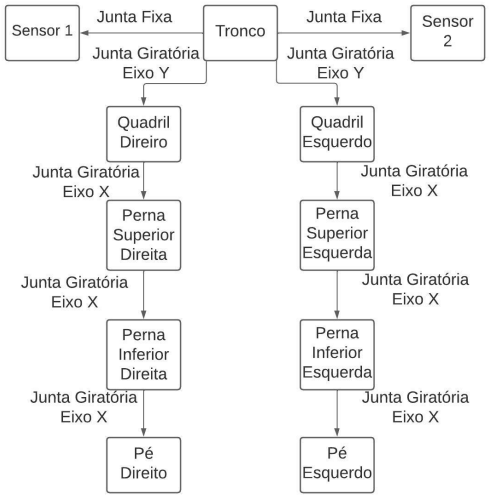
```
<joint name="nome_junta" type="tipo_junta">
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <parent link="link_pai"/>
  <child link ="link_filho"/>
  <axis xyz = "1 0 0"/>
  <limit lower="-1.57" upper="1.57" velocity="1" effort="1"/>
  <dynamics damping="10.0" friction="10.0"/>
</joint>
```

Fonte: Autoria própria

Neste formato é possível adicionar plugins capazes de interligar o modelo em URDF com outras ferrametas de simulação, como o *Gazebo*, para tanto basta apenas adicionar a tag do plugins com as características específicas dela, como qual sensor está sendo usado e o link em que ele estará posicionado (OPEN, 2014).

Para este projeto foi montado um robô bípede virtual de oito graus de liberdade, conforme descrito na Figura 3. Este modelo tem duas juntas capazes de movimentar no eixo lateral, localizadas uma em cada quadril, e seis juntas capazes de estender e contrair a perna para realizar um movimento no eixo frontal. Também foi necessário adicionar mais duas juntas fixas para os links dos sensores acelerômetros, para obtenção de dados.

Figura 3 – Diagrama Links e Juntas Robô Bípede



Fonte: Autoria própria

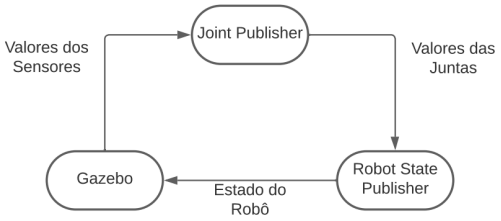
PROGRAMAÇÃO NO ROS

Para simular um robô real em ROS com Gazebo, é necessário 2 nós pricipais, um publicando a descrição do robô, que recebe os valores da junta e publica a descrição física do robô, chamado *Robot State Publisher*, e um com o *Gazebo*, recebendo a descrição física do robô e publicando os valores dos sensores e juntas (NEWANS, 2021). Para programar o robô neste formato de simulação, é possível criar um outro nó que recebe os valores dos sensores e publica os valores das juntas.

A Figura 4, demostra o funcionamento da simulação neste projeto. Cada nó apresentado

realiza uma determinada função, como dito anteriormente. O nó "*Gazebo*"é responsável pelo ambiente de simulação com a apresentação visual do robô, o nó "*Robot State Publisher*"é responsável pelo recebimento dos valores das juntas e publicação do estado do robô e o nó "*Joint Publisher*"recebe os valores de aceleração dos sensores, aplica um modelo de controle para obter os valores do centro de massa e do pé, aplica cinemática inversa para achar os valores das juntas e publicá-las.

Figura 4 – Diagrama Nós Usados no Projeto



Fonte: Autoria própria

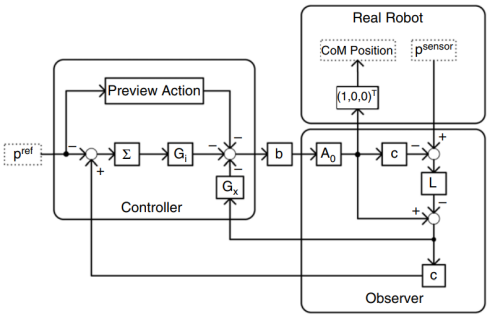
CONTROLE DO ROBÔ

Para este projeto, será aplicado um modelo de controle baseado em observador para o robô, para que seja possível gerar uma trajetória para o centro de massa e assim poder calcular os valores das juntas aplicando cinemática inversa. O modelo aplicado controla o robô a partir do ponto de zero momento. Simplificando a dinâmica de um robô bípede é possível representá-lo por um pêndulo invertido. No formato de caminhada de um corpo bípede há um ponto no chão em que o somatório das forças inerciais e gravitacionais é zero, denominado ponto de zero momento (VUKOBRATOVIC; BOROVAC, 2004), com este ponto é possível gerar um sistema confrome decrito na Equação 1.

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{z_h} & 0 & -\frac{g}{z_h} \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v \tag{1}$$

Onde x é a posição do centro de massa no eixo X, $\dot{x}$ é a velocidade do centro de massa, p é o pont de zero momento, g é a gravidade e z_h é a altura do centro de massa no eixo z. A partir do sistema apresentado na Equação 1, os autores Czarnetzki, Kerner e Urbann chegaram a uma abordagem de modelo de controle ótimo com o uso de demanda futura e um observador (CZARNETSKI; KERNER; URBANN, 2009). Com essas características, o sistema de controle pode se basear em valores esperados futuros, tornando mais fácil atingir um valor esperado, também é possível estimar a posição do robô com a presença do observador no sistema. O algoritmo final do sistema está representado na Figura 5, onde A, b e C são as matrizes do sistema em espaço de estados desenvolvidos pelos autores, L é a matriz de estimação de estado pelo observador, Gd, Gx e Gi são os ganhos da demanda futura, feedback proporcional e ação integral, respectivamente.

Figura 5 – Diagrama Algoritmo Final do Modelo de Controle

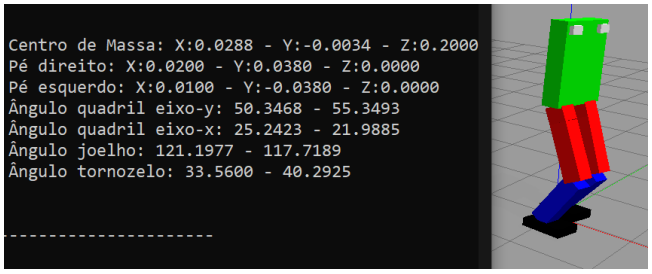


Fonte: (CZARNETSKI; KERNER; URBANN, 2009)

RESULTADOS

Executando os nós apresentados na Figura 4 foi possível simular o robô bípede conforme a proposta do projeto. A Figura 6 demonstra o design do robô no software Gazebo, juntamente com a saída no terminal contendo os valores das posições do centro de massa e dos pés e ,também, os valores dos ângulos nas juntas, em graus. É possível observar o funcionamentos das juntas e dos links conforme o arquivo URDF desenvolvido. Perante o movimento, o algoritmo de controle aplicado prove os valores das juntas que foram passados para o robô, analisando a Figura da simulação, 6, é possível observar os valores publicados pelo nó *Joint Publisher*, esses valores foram enviados ao *Robot State Publisher* e depois enviado o estado do robô para o *Gazebo*, gerando o movimento da imagem.

Figura 6 – Robô Virtual em Movimento no Software Gazebo



Fonte: Autoria própria

Com está simulação foi possível validar o modelo de controle proposto por Czarnetski,Kerner e Urbann (CZARNETSKI; KERNER; URBANN, 2009), com isso é correto afirmar que com a simulação de um sistema robótico por meio do ROS é possível avaliar sistemas de controle sem a necessidade de desenvolver a máquina fisicamente. Além do mais, com a simulação, também foi possível analisar os movimentos que o robô é capaz de realizar, concluindo que com a simulação tem-se uma forma de avaliar a dinâmica do movimento de um robô. No entanto, há divergencias na simulação que podem tornar a simulação não satisfatória. Um desses fatores é o design das peças, pois na simulação eles são formas geométricas que não colidem entre si, deferente das peças reais que se colidem, outro fator é a interação com o ambiente, como características do solo ou até influência do vento, caso seja desenvolvido uma máquina desconsiderando estes fatores é possível que está apresente problemas

de funcionamento, como mal desempenho na aplicação de uma tarefa, pois foi desconsiderado influências externas. Para desenvolver e solucionar esses problemas é necessário uma pesquisa mais aprofundada do ROS.

CONCLUSÂO

O projeto apresentado neste artigo propõe uma forma de simulação de sistemas robóticos com o auxilio de um sistema que garante uma gama de bibliotecas relacionadas ao assunto. Foi desenvolvido a modelagem de um robô real junto com a aplicação de um algoritmo de controle para testar a simulação. Os resultados satisfazem a proposta apresentada, provanto que é possível validar algumas características do robô, como controle e dinâmica, no entanto, ainda restam alguns detalhes que podem tornar a simulação não satisfatória, por exemplo influências externas.

Agradecimentos

Gostaria de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, por viabilizar o apoio financeiro ao projeto por meio da bolsa de estudos.

REFERÊNCIAS

CZARNETSKI, S.; KERNER, S.; URBANN, O. **Observer-based dynamic walking control for biped robots**. Robotics and Autonomous Systems, 2009.

MACIEL E. H.; HENRIQUES R. V. B.; LAGES W. F. **Control of a Biped Robot Using the Robot Operating System.** 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol, 2014.

NEWANS, J. **Getting Ready for ROS Part 8: Simulating with Gazebo.** [S.l.: s.n.], 2021. Acessado em: 18 de Julho de 2023. Disponível em: https://articulatedrobotics.xyz/ready-for-ros-8-gazebo/

OPEN ROBOTICS. **Building a Visual Model From Scratch.** [S.l.: s.n.], 2023. Acessado em: 12 de Julho de 2023. Disponível em: https://docs.ros.org/en/humble/Tutorials/Intermediate/URDF/Building-a-Visual-Robot-Model-with-URDF-from-Scratch.html

OPEN ROBOTICS. **ROS 2 Documentation.** [S.l.: s.n.], 2023. Acessado em: 02 de Julho de 2023. Disponível em: https://docs.ros.org/en/humble/index.html

OPEN ROBOTICS. **Setting Up a Robot Simulation (Gazebo).** [S.l.: s.n.], 2023. Acessado em: 04 de Julho de 2023. Disponível em: https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo/Gazebo.html

OPEN ROBOTICS. **Tutorial: Using gazebo Plugins With ROS.** [S.l.: s.n.], 2014. Acesso em: 18 de Julho de 2023. Disponível em: https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

VUKOBRATOVIC, M.; BOROVAC, B. **Zero-moment Point - Thirty Five Years of Its Life.** International Journal of Humanoid Robotics, 2004.