Modelagens e Simulações Computacionais Aplicadas ao Desenvolvimento Remoto de Sistemas Mecatrônicos

Getúlio A. da Silva * **Erick Nathan M. Alves** * **Miguel L. Rodrigues** * **Renato S. Dâmaso** *

* *Grupo PET Eng. Mecatrônica, Centro Federal de Educação Tecnológica de Minas Gerais, Unidade de Divinópolis, MG (E-mails: getulio.andrade1@gmail.com, ericknathancoro@hotmail.com, miguel.lukas52@gmail.com, renatosd@cefetmg.br)*

Abstract: The use of computer simulations allowed some of the research carried out by the PET group in the robotics laboratory of CEFET-MG in Divinópolis to continue during the pandemic period. As a way of recommending the use of simulators in academic activities, this article deals with some details of realistic simulators such as CoppeliaSim and the Webots. It is also shown as an example the accomplishment of a research, the development of a game of pick up and throw boxes in the Webots. The use of this simulator also allowed the continuity of both the Seeds Project and the Internal Robotics Competition.

Resumo: A utilização de simulações computacionais permitiu que algumas das iniciativas realizadas pelo grupo PET, no laboratório de robótica do CEFET-MG, em Divinópolis, pudessem continuar no período de pandemia. Como uma forma de recomendar o uso de simuladores em atividades acadêmicas, esse artigo trata sobre alguns detalhes de simuladores realísticos, como o CoppeliaSim e o Webots. Também é mostrada, como exemplo, a realização de uma pesquisa, que trata do desenvolvimento de um jogo de pegar e jogar caixas no Webots. O uso desse simulador também permitiu a continuidade tanto do Projeto Sementes quanto da Competição Interna de Robótica.

Keywords: Mobile Robotics; Remote Development; Computer Modeling; Realistic Simulators; Academic Competition.

Palavras-chaves: Robótica Móvel; Desenvolvimento Remoto; Modelagem Computacional; Simuladores Realísticos; Competição Acadêmica.

1. INTRODUÇÃO

Em tempos de pandemia do Covid-19, as instituições de ensino superior e de pesquisa viram-se diante da inesperada realidade do ensino e trabalho remotos, desafio este, proveniente de um limitado acesso aos campi, bem como aos recursos disponíveis em seus laboratórios. Nesse sentido, alguns dos desenvolvimentos, que dependiam de tais recursos, precisaram ser interrompidos, enquanto outros foram adaptados e adequados, sendo de suma importância a busca de medidas e estratégias que possibilitassem algum avanço dos trabalhos, mesmo que de forma remota e estruturalmente "limitada".

No âmbito do grupo PET Eng. Mecatrônica da unidade de Divinópolis do CEFET-MG, buscou-se informações sobre simuladores realísticos da área da robótica, que é a principal área de pesquisa do grupo. Como é tradição do grupo, a organização de competições acadêmicas de robótica no escopo de seu Projeto Sementes (PS) e da Competição Interna de Robótica (CIR) ocorreram no formato físico presencial nos anos de 2017, 2018 e 2019 (Dâmaso, 2020). No entanto, com o advento inesperado da pandemia, seria razoável o levantamento e validação de alternativas que viabilizassem a continuidade dessas iniciativas, mesmo que no formato virtual, compatível com a necessidade de distanciamento social.

Este artigo tem por objetivo, mostrar o uso do simulador Webots no desenvolvimento dessa iniciativa de ensino e extensão, bem como em uma das pesquisas realizadas pelo grupo, o "jogo de pegar e jogar caixas" com o robô Kuka youBot, como uma forma de registrar essa aplicação e para, possivelmente, incentivar a utilização desse tipo de simulador por outras instituições, que queiram se beneficiar de uma maior disponibilidade e custos reduzidos para o ensino de programação e robótica, de uma forma interativa, realística e eficiente. São apresentados também, aspectos relacionados à modelagem e implementação de um robô móvel, em dois ambientes de simulação, para auxiliar na introdução ao Webots.

2. SIMULADORES REALÍSTICOS

O grupo intensificou a pesquisa acerca de simuladores da área da robótica, que fossem de propósito geral ou educacional. Além disso, diferentemente de alguns simuladores como o RoboAnalyzer e o RoboDK, era de suma

importância que o simulador fosse realístico, ou seja, que incorporasse a modelagem física (modelagem mecânica e dinâmica mais precisa, possibilidade de colisões e influência de fatores externos reais, como a aceleração da gravidade, atrito e arrasto, por exemplo), como ocorre nos simuladores CoppeliaSim, Webots e Gazebo. Em um simulador realístico como esses, se um objeto for agarrado e arremessado por um robô, ele irá cair sob a ação da gravidade e colidir com o chão ou com outro objeto da simulação, sendo as reações dos objetos e do "mundo" (ambiente virtual criado na simulação) dependentes dos materiais envolvidos nas interações.

Como o simulador Gazebo, juntamente com o pacote ROS (*Robotic Operating System*) vem sendo usado há pouco tempo pelo grupo, ainda restrito a implementações de robótica industrial, e sua interface e operação serem mais complexas para os usuários e participantes das iniciativas propostas, ele não foi escolhido para a adequação das competições à modalidade virtual.

2.1 Simulador CoppeliaSim

Chamado de V-Rep até 2019, esse simulador de propósito geral disponibiliza uma variedade de robôs móveis (como o minirobô e-puck) e não móveis. Além disso, ele permite a importação de elementos modelados num ambiente 3D, como o SolidWorks, por exemplo, a montagem, motorização e adição de novos robôs, a serem programados em LUA, C, C++, etc (Rohmer et al., 2013).

A Figura 1 mostra a modelagem parcial do robô móvel a rodas do Projeto Sementes (Dâmaso, 2020), já que está faltando a base superior, o LED RGB e o copo plástico translúcido, destinado a destacar e tornar a iluminação produzida pelo LED mais suave e difusa.

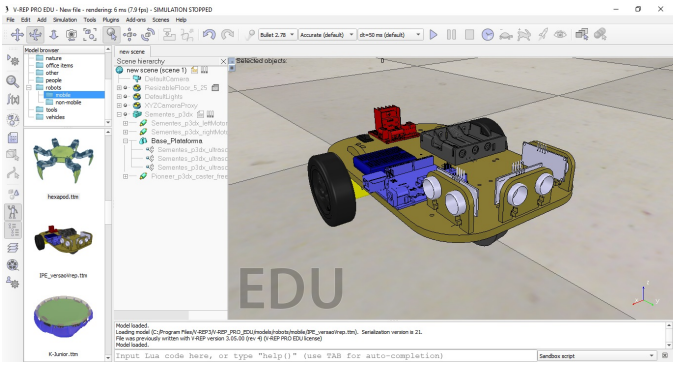

Figura 1. Modelagem do robô do PS no CoppeliaSim

2.2 Simulador Webots

A opção por este software deu-se também, devido à grande quantidade de objetos nativos, já modelados, como: pisos, blocos, caixas, peças, robôs, etc; a facilidade de importar objetos feitos em programas de modelagem 3D (como o SolidWorks), a inclusão de propriedades físicas, como características de materiais, limites dos objetos, tipos de colisões, motorização, etc. Além disso, outra característica do Webots que contribuiu para a sua utilização frequente pelo grupo PET foi a possibilidade de programar os elementos ativos nas simulações com linguagens de propósito geral, como C, C++, Python, entre outras.

Criado e mantido como o principal produto da empresa suiça Cyberbotics Ltd. desde 1998, o Webots é um software de simulação 3D realístico de código aberto usado no meio industrial, educacional e de pesquisa. Além do grande número de objetos estáticos e dinâmicos, como diversos robôs, controlados por funções de bibliotecas nativas, o software permite a modelagem de novos componentes e suas respectivas programações, corroborando para um ambiente de desenvolvimento e simulações completo e poderoso (Cyberbotics, 2022).

A Figura 2 mostra a interface do Webots, sendo seu ambiente de desenvolvimento, composto por quatro partes principais: o Visualizador 3D, que é onde se observa toda a criação, comportamento e funções que o usuário está aplicando na simulação, sendo estes, desde as configurações físicas e mecânicas do solo, ao objeto que está sendo interagido; a Árvore de Cenas, onde é possível produzir resultados diversos utilizando-se de nós e sub nós, desde a modificação das interações de um nó pai com um nó filho, até o movimento de toda uma peça ou montagem complexa; o resultado de todas as alterações podem ser validadas no Visualizador 3D; o Controlador, que permite programar em diversas linguagens, para serem aplicados aos nós da árvores de cenas e permitir que determinadas peças se comportem conforme programado, sejam estas dinâmicas, ou realizando alguma função de medição de velocidade, distância ou qualquer outra aplicação que o usuário deseje; e por fim, a janela Console, que permite observar desde erros cometidos no controlador, até erros decorrentes de se utilizar um nó incorretamente, por exemplo. Os recursos de depuração do Webots são limitados até então (versão R2022a), sendo comum a utilização do próprio terminal para acompanhar as leituras dos sensores, valores de variáveis e andamento do controlador.

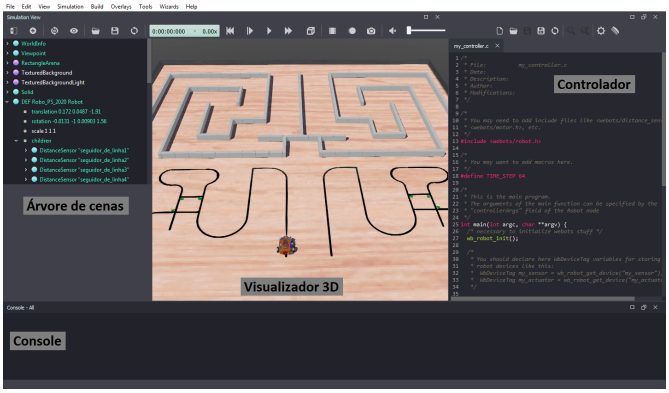

Figura 2. Ambiente de desenvolvimento do Webots

É na Árvore de Cenas que o software pode realizar importações e exportações de peças, havendo dois tipos de importações diferentes. A primeira é utilizada para arquivos que foram exportados diretamente pelo próprio aplicativo Webots, havendo a necessidade de utilizar determinada peça em outro mundo criado pela aplicação. Esta é realizada quando o usuário acessa a opção de criar um novo nó, e sem que qualquer nó seja criado ele decida importar alguma peça já presente em seu computador. Para isso os tipos de arquivos aceitos pela importação são wbo, 3ds, bvh, blend, dae, fbx, stl, wrl, obj e x3d. Já a segunda maneira é a importação de modelos 3D na própria área de

arquivos do software, sendo possível importar arquivos de diversas extensões de modelagem 3D, oriundos de softwares de CAD, tal como o SolidWorks (Systemes, 2022). Esta opção possui uma variedade maior de arquivos aceitáveis. É possível exportar arquivos do Webots, mas, diferentemente da importação que restringe a variedade de arquivos possíveis, a exportação possui algumas terminações a mais para serem exportadas, possibilitando assim a importação por outros programas de simulação.

3. MODELAGENS E SIMULAÇÕES

A modelagem tanto de robôs, quanto de diversos outros componentes desejáveis ao utilizador do Webots, é feita através da Árvore de cenas. Neste campo, trabalha-se os nós (*nodes*) disponíveis pela aplicação, e com a interação entre os pontos, é possível dar origem a diferentes formas, simples ou complexas, capazes de realizar funções de maneira estática ou móvel.

Os principais tipos de nós utilizados são o *Solid* e o *Transform*. A partir deles, nós especiais como o *robot*, que possui *campos (fields)*, apropriados às devidas configurações para a montagem de um robô, ou *Shape*, permitindo a construção de formas geométricas dentro do mundo, podem ser devidamente utilizados. Diversos outros nós também estão disponíveis para a utilização, mas suas bases são compostas de nós *Transforms*, que são a forma mais básica de um nó, possuindo campos que permitem apenas a movimentação básica de uma peça, bem como o controle de escala e a capacidade de possuir outros nós dentro de si mesmo, no *field* chamado *children*. O nó *Solid* é um nó que possui todos os campos do nó *Transform*, com o acréscimo de outras funções importantes, principalmente a definição de um contorno para o objeto em questão, o chamado *boundingObject*. Este parâmetro é o que define os limites de colisão de cada parte modelada, possibilitando que o objeto interaja física e mecanicamente com o resto do mundo. Por fim, há o campo *physics*, que adiciona características física ao objeto. Este conjunto de nós e configurações transformam um objeto, até então estático e fisicamente indefinido, em um objeto sólido, com possível interação com o meio virtual em que está inserido.

Alguns exemplos de projetos criados usando o Webots são exemplificados a seguir. Nota-se que a criação da grande maioria das partes mais delicadas, ou que possuem mais detalhes, foram realizadas em softwares externos, especializados em modelagem tridimensional, como é o caso do SolidWork, que já foi apresentado. Com isso, apesar da modelagem ser possível diretamente no Webots, há uma certa dificuldade em elaborar formas mais complexas dentro do software, sendo assim mais viável, a modelagem externa e a posterior importação no Webots. Desta maneira, o simulador interpreta o objeto importado, como um conjunto de coordenadas tridimensionais, que juntas formarão o corpo modelado.

3.1 Modelagem do Robô Móvel do PS

Ao longo dos três anos de desenvolvimento e utilização do robô móvel de baixo custo baseado em Arduino, tanto no PS quanto na CIR, foram montadas mais de 70 unidades, a um custo de cerca de 230 reais cada uma. Esta experiência e familiaridade com o protótipo, certamente facilitou a modelagem do mesmo, para modelagem e uso posterior, em simulações.

Um robô de tração diferencial, como o robô móvel do PS, é um sistema dinâmico composto por duas rodas comuns, acopladas a motores com redução individual, além de uma roda de apoio, não motorizada. Usualmente, o modelo cinemático deste sistema é obtido através da integração numérica das velocidades atuantes sobre o mesmo. Para isso, estas são decompostas no frame inercial de referência, e então, integradas a cada intervalo de tempo dt suficientemente pequeno, como mostra a Equação (1).

$$\int_0^t \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} dt = \int_0^t \begin{bmatrix} cos(\theta_t) & 0 \\ sin(\theta_t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \quad (1)$$

Onde V é a velocidade *linear*, ω a velocidade *angular* e θ o ângulo de orientação do robô com relação ao frame inercial de referência.

Com o modelo cinemático do sistema em mãos, o processo de simulação e projeto de controladores torna-se simples e computacionalmente barato, devido ao fato de não necessitar de um ambiente de simulação 3D. É importante ressaltar que o modelo cinemático não leva em consideração as forças atuantes sobre o sistema e, por isso, contém incertezas em relação ao sistema real.

No modelo do robô do PS, como pode ser visto na Figura 3, foram implementados: três sensores de ultrassom, dispostos em sua porção frontal; dois encoders para as rodas atuadas; e cinco sensores de luz, ou LDR's - *Light Dependent Resistors* (que no Webots, foram adaptados para sensores de distância, por conveniência) para a placa do seguidor de linha em sua parte inferior. Há também um LED RGB em sua parte superior, sob o copo opaco, destinado a destacar a luz emitida pelo LED, além de torná-la mais difusa.

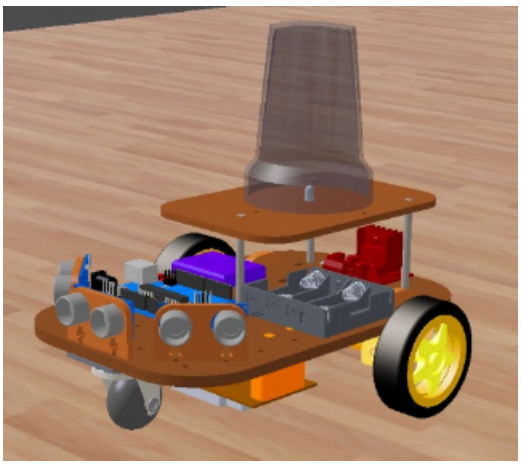

Figura 3. Robô do PS modelado no Webots.

A modelagem do robô utilizado foi feita no software SolidWorks. As peças foram importadas, uma a uma, para o simulador Webots, montadas com o emprego da estrutura da Árvore de Cena, motorizadas e os sensores foram instalados. Dessa forma, foi possível comparar o modelo matemático com o robô simulado, obtendo resultados pró-

ximos, uma vez que o modelo cinemático não leva em conta as forças atuantes sobre o robô.

Para a construção detalhada do dispositivo considerado, cria-se um nó-pai *Robot*, e nele é inserido um nó *Group*, carregando diversos nós *Transform* representando cada peça que não possui função especial no aparelho. Isso é, basicamente uma carcaça para o dispositivo. Foi criado também um segundo *Group* contendo nós *Transform* que possuíam nós-filhos, também do tipo *Transform*, uma vez que, os nós que possuem nós-filhos *Transform* não podem ter um contorno definido, impedindo que a carcaça do robô fosse perfeitamente formada. Em seguida, os três sensores frontais foram colocados em suas respectivas posições, sendo utilizados para isso os nós *DistanceSensor*, que como o próprio nome diz, fazem a função de sensores de distância e permitem que o Robô perceba a distância que ele está de algum objeto na linha de ação desse nó. Os cinco nós sob o robô foram posicionados, também como nós *DistanceSensor*, mas esses possuem seus campos modificados em *type "infra-red"* ao contrário do *generic* dos sensores frontais, e em *numberOfRays: 9* ao contrário do raio único dos sensores posicionados na frente. É esta quantidade de raios que proporcionam uma variação na leitura das distâncias entre os diferentes sensores na parte de baixo do carrinho, permitindo posicioná-lo com mais facilidade sobre a linha que o robô deve seguir.

Para a criação das rodas foi usado o nó *HingeJoint* que, com a configuração do campo certo e adição de um *RotationalMotor* no campo *device*, permite que o *Solid* adicionado no campo abaixo, gire em uma direção da escolha do usuário. Esta configuração foi feita para as três rodas. Um importante ponto a salientar é a configuração da ancora nos parâmetros do nó, tendo como referência a translação do sólido escolhido para efetuar o giro. Como último nó especial, foi adicionado o *LED* e então, foi programado no campo específico, as cores que ele iria assumir. As rodas, tanto traseiras quanto a própria carcaça do robô tiveram seus campos *boundingObject* e *physics* ligados aos respectivos nós, garantindo assim, que o robô móvel tenha seus limites de colisão bem definidos. Assim, a Árvore de Cena final do robô móvel pode ser visualizada na Figura 4.

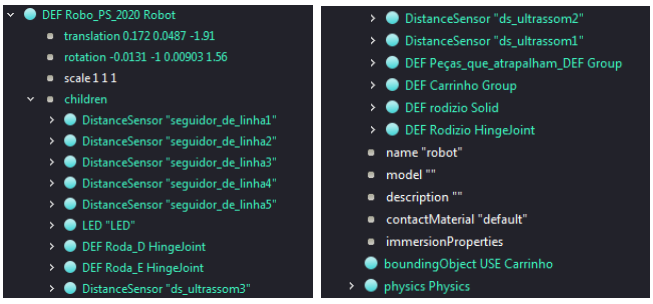

Figura 4. Árvore de cenas do Robô do PS 2020

3.2 Modelagem das Arenas de Competição

A arena da CIR 2020, conforme Figura 5, era dupla e espelhada, nos moldes da arena física do PS e da CIR em 2019. A tarefa proposta para a CIR 2020 foi composta por um primeiro trecho de seguidor de linha, seguindo as regras na modalidade IEEE Rescue, da Olimpíada Brasileira de Robótica (OBR, 2020), especialmente em relação às regras de sinalização. Essa primeira parte era seguida por um trecho de navegação em labirinto, sendo que uma de suas paredes se movia de forma randômica entre três posições. Essa modificação e comportamento aleatório de uma parte do labirinto seria, de certa forma, um incentivo ao uso de um comportamento reativo, por parte das equipes, na programação de seu robô.

Para a configuração dessa movimentação randômica, foi utilizado um nó *Robot*, e em seu *children* foi inserido um *Receiver*, que tem como função receber pacotes de dados emitidos por nós emissores. Em seu campo *channel* é enumerada a quantidade de entidades que serão atribuídas uma configuração especial, neste caso, a movimentação randômica entre três posições. No nó *Robot*, o campo *Supervisor* é alterado para *TRUE*, permitindo o controle de parâmetros dos elementos da simulação. Assim, foi desenvolvido um controlador no nó para gerenciar os movimentos dos blocos.

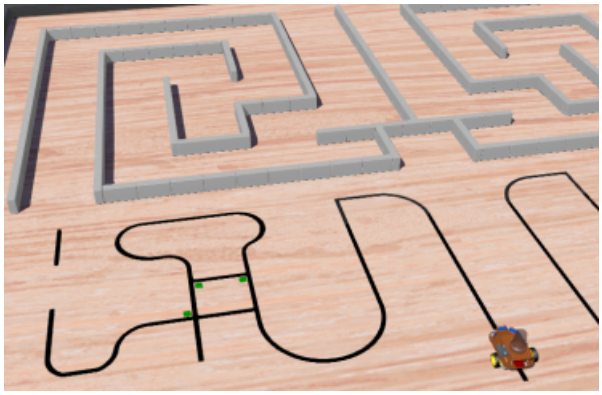

Figura 5. Vista parcial da arena da CIR 2020 no Webots.

Na Figura 6 observa-se a arena da CIR 2021. Nessa referida edição, o grupo PET Eng. Mecatrônica decidiu reforçar os conceitos e aprendizados adquiridos nos treinamentos ministrados pelo grupo e disponibilizados em diversos vídeos, através do YouTube (PET, 2021). Os desafios de seguidor de linha e labirinto foram mantidos, entretanto, com a presença de um circuito bastante longo, inspirado no traçado do autódromo de Interlagos em São Paulo. Nesse trajeto foram incluídas duas armadilhas, uma para cada linha, de forma a testar a programação dos sensores de identificação da sinalização. Outrossim, para o trecho do labirinto, o grupo implementou um formato circular, sendo seu nível mais central também dotado de movimento de rotação, com velocidade e intervalos de parada variáveis e sortidos randomicamente, como mostrado no vídeo indicado acima. Diferentemente do proposto em 2020, o labirinto unificado, ou seja, compartilhado pelos dois robôs competidores, simultaneamente. Nova condição esta, que deixaria a competição mais direta e emocionante, contudo, que exigiria alterações nas regras. As colisões consideradas intencionais eram punidas uma advertência (cartão amarelo), ou com a exclusão (cartão vermelho) caso não fossem corrigidas em rodadas futuras da competição. Outro detalhe do labirinto estava em sua saída, que tinha uma pequena parede seletora, que se movia ao detectar a passagem do primeiro robô, direcionando assim o segundo robô para a outra linha do seguidor.

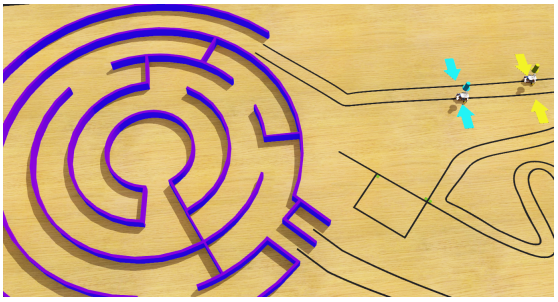

Figura 6. Vista parcial da arena da CIR 2021.

No caso das modelagens das arenas de competição e do robô móvel do PS, a sequência básica foi a mistura da escolha e configuração de elementos já disponíveis no Webots com elementos salvos em arquivos .DWG, .WRL, .STL e .OBJ, importados para o simulador e montados com juntas, motores, sensores, etc. Para muitos dos elementos estáticos, componentes do cenário, bastava posicioná-los e configurar suas características físicas e aparências. Já para os elementos dinâmicos, deve-se atentar ainda, a parâmetros como o *bounding object* (limites de colisão dos objetos), massa, fricção e limites angulares das juntas, orientação e propriedades de sensores e atuadores, dentre outros aspectos que definem como estes elementos móveis se comportarão na simulação. Além disso, há a necessidade de um controlador para que o funcionamento dos elementos dinâmicos seja programável e definido.

3.3 Modelagem do Jogo de Pegar e Jogar Caixas

Dentre as atividades desenvolvidas pelo grupo PET Eng. Mecatrônica, a pesquisa sobre "Estrategias de Controle com Inteligência Artificial para a Tarefa de Pegar e Jogar Caixas entre Manipuladores Moveis" foi um dos temas iniciados graças ao uso de um simulador realístico. A ideia dessa pesquisa consistia no uso de modelos de execução de tarefas em robótica cuja realização se faz de modo autônomo. Neste contexto, robôs capacitados para o transporte de carga tomam decisões de direcionamento e realização de tarefas de forma autônoma, sem que haja a intervenção humana. Para isso, são utilizados modelos de computação baseados em algoritmos genéticos e redes neurais.

Na pesquisa em questão, foi utilizado também, o ambiente de simulação Webots, apresentado anteriormente neste artigo. O robô selecionado foi o holonômico Kuka youBot, que pode ser visto na Figura 7, sendo este composto por uma plataforma móvel de movimentação, além de quatro rodas omnidirecionais do tipo Mecanum, além de um manipulador com 5 GDL's (Graus de Liberdade) e uma garra com dois dedos paralelos integrada (Cyberbotics, 2022).

O cenário modelado é semelhante ao tênis de mesa, utilizando uma mesa com uma divisória rígida no meio, além de uma pequena caixa de madeira, como pode ser visto na Figura 7. Os objetos que constituem o cenário são nativos ou foram modelados no próprio simulador. A tarefa proposta é que o robô conseguisse, através de um controlador baseado em redes neurais, pegar e arremessar a caixa para o outro lado da mesa, demarcada pela divisória central e, em sequência, que outro robô pegasse a caixa e arremessasse de volta. O ponto terminaria quando a caixa não passasse ao outro lado ou caísse fora da mesa.

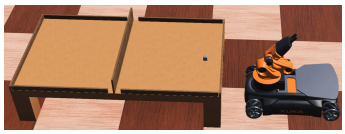

Figura 7. Cenário do jogo de Pegar e Jogar com o *youBot*

O Webots oferece a possibilidade de programar controladores interna ou externamente ao simulador. Nesse caso, optou-se por desenvolver o controlador de forma externa, utilizando a API que o simulador disponibiliza. A implementação do controlador foi feita na linguagem Python, tendo como referência o controlador de exemplo, que acompanha o robô Kuka youBot no simulador. A IDE utilizada foi o PyCharm (versão gratuita) da JetBrains.

A API disponibilizada pelo Webots contém funções que permitem a obtenção de informações sobre os objetos que estão sendo simulados, como coordenadas, orientação, velocidade, etc. Dentre as funções disponíveis, foram utilizadas, com maior frequência, as de obtenção da posição e orientação dos objetos, sendo esta última, na convenção eixo-ângulo (ângulos de Euler), ou seja, a transposição de dois sistemas inerciais de eixos tridimensionais: um fixo no *frame* inercial de referência e outro que gira junto ao corpo do objeto. Dessa forma, além do controlador, foi desenvolvida uma biblioteca que encapsula estas e outras funções, a fim de tornar a obtenção dos dados uma tarefa menos repetitiva. Esta biblioteca foi, posteriormente, integrada ao controlador.

Após a implementação da biblioteca para obtenção das informações acerca dos objetos em simulação, deu-se início ao processo de tratamento dos dados, definindo e implementando classes de acordo com o tipo de dado a ser tratado. As informações de posição e orientação podem ser representadas por matrizes e vetores, implementados no controlador, por exemplo. Os valores de posição representam a translação do *frame* atual do robô, em relação ao *frame* inercial de referência, podendo também ser representados por um vetor. Por fim, a orientação é representada por valores de rotação em torno de cada eixo, sendo estes: *Roll* para o eixo X, *Pitch* para o eixo Y, e *Yaw* para o eixo Z. Esses, por sua vez, são representados na convenção eixo-ângulo, a qual define os valores nx, ny, nz, e α. Os três primeiros valores representam um vetor normalizado que aponta em direção ao eixo que a rotação deve ser realizada. Já o quarto valor, α, especifica o ângulo de rotação em radianos em torno do eixo (Spong et al., 2005).

Durante a realização de testes com o modelo do robô Kuka youBot, foi observado que os valores de orientação não eram coerentes com o que se observava, como é perceptível na Figura 8.

Lema *1.* Em um sistema ortonormal, tem-se que a matriz inversa de uma matriz de rotação é o mesmo que a transposta dessa matriz de rotação. (Spong et al., 2005)

Pesquisando sobre o problema, notou-se que o robô possui uma composição de rotação em torno do eixo X de -90 graus, comprometendo assim a obtenção do valor de rotação em torno do eixo vertical, sendo este detalhe,

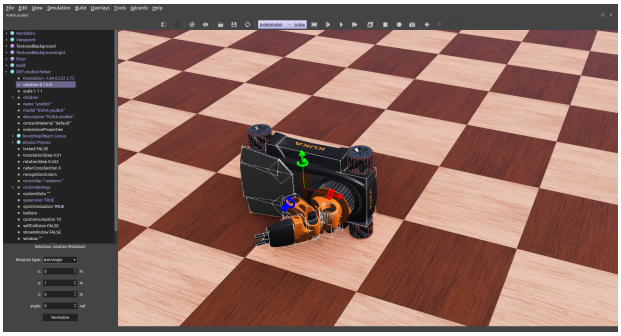

Figura 8. Rotação inesperada do *youBot* no eixo X

crucial para a realização da tarefa proposta. Assim, para corrigir o impasse e obter os valores corretos de orientação, é possível, através do Lema (1), obter a matriz de rotação, compensando a rotação em X, através de (2) que, por meio do Lema (1), origina (3).

$$R_{x,-90} \cdot R_{z,\alpha} = R \tag{2}$$

$$R_{z,\alpha} = R \cdot (R_x, -90)^T \tag{3}$$

A matriz R é obtida através dos valores de orientação do robô nx, ny, nz, e θ. Segundo (Spong et al., 2005), o cálculo da matriz de rotação em relação ao eixo X, de um ângulo θ, é dada por (4).

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \tag{4}$$

Substituindo θ por $-90°$, obtem-se (5).

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5}$$

De acordo com Kevin M. Lynch (2017), utilizando a fórmula de **Rodrigues** para rotações é possível obter a matriz R, por meio de (6).

$$R(\hat{\omega}, \alpha) = e^{[\hat{\omega}]\alpha} = I + sin\alpha[\hat{\omega}] + (1 - \cos\alpha)[\hat{\omega}]^2 \tag{6}$$

Onde $[\hat{\omega}]$ é o vetor unitário dado por nx, ny, e nz e α é o ângulo de rotação, em radianos, em torno desse vetor. Por fim, através de técnicas de cinemática inversa, o ângulo de rotação no eixo vertical ϕ é calculado e expresso por (7).

$$\phi = atan_2(R_{21}, R_{11}) \tag{7}$$

Essas correções foram implementadas e possibilitaram prosseguir com a tarefa de pegar a caixa. Outros procedimentos foram empregados para automatizar a tarefa de direcionar adequadamente o youBot e jogar a caixa para o outro lado da mesa. A melhoria dessa parte da pesquisa encontrava-se em desenvolvimento no período da escrita deste artigo. Como complemento das imagens presentes no artigo, foi produzido um vídeo, que pode ser visualizado em www.youtube.com/watch?v=PcsrI3WJG9Q .

4. CONCLUSÕES E PERSPECTIVAS

O PS é uma iniciativa do grupo PET Eng. Mecatrônica oferecida de forma totalmente gratuita para as escolas públicas de nível médio de Divinópolis e região, sendo de periodicidade anual. O projeto compreende as etapas de divulgação, treinamento e competição acadêmica. Apesar da modalidade virtual do PS ter sido ofertada às escolas públicas de nível médio de Divinópolis e dez escolas terem participado do treinamento, a competição acabou não acontecendo em 2020, pois apenas duas equipes efetivaram suas inscrições. Já em 2021, somente três escolas se inscreveram para o treinamento, o que inviabilizou sua realização. Pelo retorno de alguns diretores e professores multiplicadores, o principal motivo da baixa adesão, foi a conturbada transição do ensino remoto para o presencial; justificativa esta, considerada razoável e plausível, uma vez que, este retorno ao ensino presencial era prioritário e exigia um grande esforço dos docentes, principalmente devido ao fato das turmas terem sido divididas, em alguns casos, em três para cumprimento das exigências oficiais de afastamento.

Por outro lado, o fato da CIR ter acontecido em 2020 e 2021 foi considerado bastante positivo. Em 2020 foram 9 equipes inscritas, todas do curso de Eng. Mecatrônica do CEFET-MG de Divinópolis. Já em 2021, alcançou-se uma expansão para os cursos Técnico de Informática, Técnico em Mecatrônica da unidade e Graduação em Eng. de Computação, totalizando 23 equipes inscritas para a competição.

Neste ano de 2022, o planejamento é de melhorar o robô móvel com a implementação do ESP32 no lugar do Arduino Uno, oferecer novamente a modalidade presencial do PS e as modalidades virtual e presencial para a CIR, de forma a consolidar cada vez mais estas iniciativas.

Com o retorno ao ensino presencial, no início do ano, a ideia foi prosseguir com algumas pesquisas utilizando simuladores. Já outros desenvolvimentos deverão ser complementados pelo uso de recursos disponíveis no laboratório de robótica. O importante é que não se perca o investimento de tempo e conhecimento adquirido no uso de simuladores realísticos, usufruindo de suas possibilidades e benefícios à robótica móvel.

AGRADECIMENTOS

Agradecemos à Diretoria de Graduação do CEFET-MG pelo apoio financeiro ao grupo PET Eng. Mecatrônica. Agradecemos também aos participantes do grupo que contribuíram com as iniciativa apresentadas.

REFERÊNCIAS

Cyberbotics (2022). Webots. URL `http://www.cyberbotics.com`.

Dâmaso, R.S. (2020). Projeto sementes: Uma introdução à programação e à robótica para escolas públicas de nível médio. Anais do CBA, Santa Maria, RS.

Kevin M. Lynch, F.C.P. (2017). *Modern Robotics Mechanics, Planning, and Control*. Cambridge University.

OBR (2020). Olimpíada brasileira de robótica. URL `https://www.obr.org.br`.

PET (2021). Treinamento para a 7ª competição interna de robótica. URL `https://youtube.com/playlist?list=PLCJAnLcv0y_KnuKYPuYEApqRy07WlkRaQ`.

Rohmer, E., Singh, S.P.N., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ - ICIRS*, 1321–1326.

Spong, M.W., Hutchinson, S., and Vidyasagar, M. (2005). *Robot Modeling and Control*. Wiley. 1st edition.

Systemes, D. (2022). Solidworks. URL `https://www.solidworks.com`.