Análise dos principais simuladores de robótica para aplicação em Deep Reinforcement Learning para otimização de trajetória focada em energia

Rodrigo Santos Silveira [1] (rodrigosantossilveira@outlook.com), Eduardo José Lima II [2] (ejlima2@gmail.com)

[1] Universidade Federal de Minas Gerais (UFMG); Programa de Pós-Graduação em Engenharia Mecânica
[2] Universidade Federal de Minas Gerais (UFMG); Departamento de Engenharia Mecânica

RESUMO: *O mercado de robôs industriais cresce de forma acelerada em todo o mundo, e com ele a oportunidade econômica e ambiental presente na otimização de trajetória para redução do consumo de energia desses robôs. Os métodos que historicamente têm mais sucesso para esse problema são computacionais e há uma grande oportunidade para emprego de Deep Reinforcement Learning nesse tipo de problema. A construção de um ambiente de simulação para o aprendizado da rede neural, entretanto, tem várias complicações. O presente trabalho analisa as principais alternativas de simuladores de física para construção do ambiente de simulação para determinação da melhor alternativa para o problema em questão. Serão analisados aspectos construtivos dos simuladores, velocidade de execução, recursos de simulação em robótica, recursos de aprendizado de máquina presentes em cada um e integração com outras bibliotecas e módulos necessários e como cada um desses critérios afeta o objetivo final para definir qual a melhor alternativa.*

PALAVRAS-CHAVE: *Deep Reinforcement Learning, Ambiente de aprendizado, Simulação em Robótica, Otimização de Energia, Manipuladores Robóticos*

SIMULATION ENVIRONMENT ANALYSIS FOR ENERGY-EFFICIENT TRAJECTORY OPTIMIZATION IN INDUSTRIAL ROBOTICS USING DEEP REINFORCEMENT LEARNING

ABSTRACT: *The global surge in the industrial robotics sector has generated a need to enhance energy efficiency, thereby fostering economic growth and addressing environmental concerns through trajectory optimization. While traditional computational methods have exhibited commendable outcomes, the potential of Deep Reinforcement Learning (DRL) in this context remains underexplored. This study delves into the realm of DRL by focusing on its application for energy-efficient trajectory optimization. To facilitate this, an intricate simulation environment is essential for neural network training. Consequently, this research conducts a comprehensive analysis of leading physics simulators, evaluating crucial factors including design attributes, execution speed, robotic simulation capabilities, machine learning functionalities, and compatibility with requisite libraries and modules. By meticulously examining the influence of each criterion on the overarching objective, we identify the optimal simulation environment that effectively balances computational efficiency and useful resources. Through this systematic exploration, we aim to unlock the full potential of DRL in energy-efficient trajectory optimization within the realm of industrial robotics.*

KEYWORDS: *Deep Reinforcement Learning, Learning enviroment, Trajectory optimization, Robotic Manipulators, Simulation*

1. INTRODUÇÃO

O mercado de robôs industriais, já muito significativo, atingindo 14,1 bilhões de dólares em 2021 segundo a Fortune Business Insight, cresce em um ritmo acelerado, atingindo a marca de 11% de crescimento em instalações de novos robôs ao ano entre 2014 e 2018 segundo o sumário anual World Robotics 2020. Essa tendência atraiu interesse para o desenvolvimento de pesquisas. O consumo energético dos robôs industriais na indústria automobilística é em média 8% do total levantando um grande potencial de economia. Em uma grande fábrica com elevado grau de automação a energia consumida é comparável à de uma cidade média (MEIKE; RIBRICKS, 2011).

A trajetória é um dos fatores mais importantes que influenciam no consumo de energia dos robôs industriais. Além disso, sendo uma forma de melhoria via software, tem o potencial de impactar muito mais rápido a indústria e os 2,7 milhões de robôs industriais que existem globalmente segundo o sumário anual da World Robotics 2020.

O consumo de energia ao longo da trajetória depende de vários fatores, tais quais velocidade, aceleração, torque, momento de inércia, comprimento da trajetória, e hoje as soluções analíticas/teóricas não conseguem construir um modelo de consumo de energia preciso o suficiente para essas aplicações. Para além disso esses métodos não endereçam a diferença de consumo ao longo da vida útil do robô e tem dificuldade de ser generalizados para a ampla gama de modelos diferentes que existem hoje no mercado (JI; WANG, 2019).

Modelos de Machine Learning (aprendizado de máquina) e inteligência artificial têm sido usados com sucesso para resolver vários tipos de problemas de engenharia devido à sua grande capacidade de aprender padrões a partir de dados. Modelos tradicionais demandam grandes volumes de dados, mas modelos de Deep Reinforcement Learning (aprendizado por reforço com redes profundas) têm a capacidade de aprender por experiência própria a partir de uma situação real ou simulação, o que reduz as exigências para a otimização.

Simulações de robôs industriais são bem estabelecidas para planejamento de trajetórias e permitem que o modelo realize ações e aprenda com elas a uma frequência muitas ordens de grandeza superior ao que seria possível utilizando um robô real. Transformar essa simulação em um ambiente de Deep Reinforcement Learning entretanto tem vários desafios. O presente trabalho aborda a primeira etapa na otimização de consumo de energia de manipuladores robóticos via reinforcemente learning, a construção do ambiente de aprendizado consolidando a simulação do robô e o ambiente de aprendizado e a função de recompensa, ações, rede neural e atualização da política.

2. MATERIAIS E MÉTODOS

Ambientes de aprendizado em Deep Reinforcement Learning são objetos computacionais que devem, dado um espaço de ações possíveis, receber uma ação realizada pelo agente e retornar o

estado do agente após o resultado daquela ação, bem como a recompensa atribuída a ela. Ao final do episódio, seja porque o objetivo foi alcançado ou o número máximo de passos usado, o ambiente também deve retornar a recompensa final acumulada, que será fundamental no processo de aprendizado e atualização da política por parte do agente. No contexto de uma simulação de consumo de energia de um manipulador robótico não é difícil imaginar o objetivo final como sendo alcançar a posição final comandada e a recompensa como dependente da quantidade de energia total utilizada. Dependente porque é a recompensa que será otimizada e, se ela não envolver atingir o ponto final, o agente encontrará formas de burlar o objetivo proposto de modo a maximizar a recompensa obtida.

A maioria dessas características é perfeitamente possível na maior parte dos simuladores de robótica, mas mais que ser possível, o quão fácil elas são de se obter importa. Uma forma muito comum de se atualizar a política de agentes de Deep Reinforcement Learning é através de estratégias evolucionárias com uma biblioteca conhecida como petting zoo. Nessas estratégias uma população de vários agentes diferentes executa a tarefa ao mesmo tempo para que possa ser pontuada e selecionada, servindo de base para a próxima geração. Nesses casos principalmente a eficiência de execução da simulação e de conexão com os outros módulos e bibliotecas é de suma importância.

Simuladores de robótica são muito utilizadas em pesquisa. O fato de permitirem acesso a uma grande variedade de plataformas robóticas a um custo muito baixo ou inexistente e sem a preocupação de danificar ou desgastar peças caras os torna muito úteis. Quando se fala em aprendizado de máquina, em que o volume de dados é essencial, a capacidade de simulação rápida e em paralelo os torna praticamente obrigatórios para a maioria das aplicações.

Dentro da grande variedade de ferramentas de simulação de robótica existem várias subcategorias, seja para robótica móvel, manipuladores robóticos, robôs aquáticos ou aéreos, robôs médicos e soft robotics (robótica mole). É importante dizer que cada uma tem suas particularidades e a melhor ferramenta para uma não necessariamente é para as outras. Aqui serão analisadas as melhores ferramentas para manipuladores robóticos. (COLLINS et al, 2021)

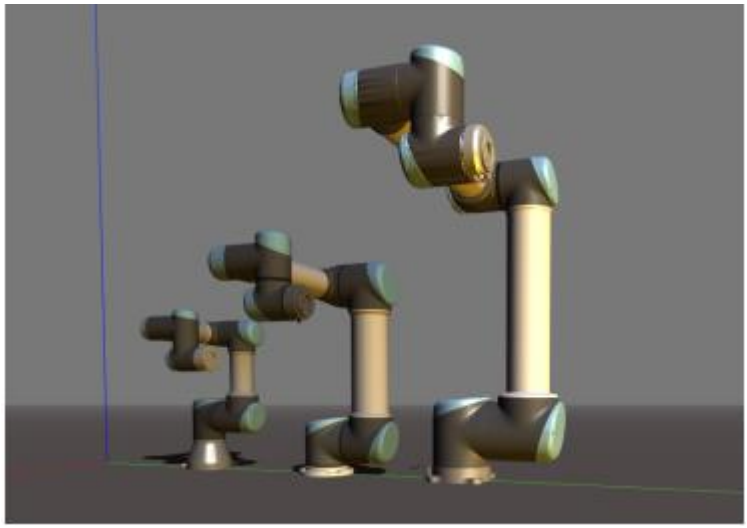
Nessa categoria existem diversas ferramentas proprietárias, com a maioria dos fabricantes tendo a sua própria, mas também algumas ferramentas abertas com bastante potencial, como é o caso do Robotics Toolbox for Python e do Pybotics.

Pybotics foi construída como uma versão em python para a versão em matlab do robotics toolbox, muito utilizada dentro da academia mas não muito acessível fora dela. O simulador funciona apenas com a notação de Denavit-Hatenberg e é todo escrito em python, aproveitando-se de bibliotecas reconhecidas e relativamente eficientes como Numpy e dependendo de conexões externas para uso em aprendizagem de máquina. (NADEAU, 2019)

O Robotics Toolbox for Python é uma das ferramentas de simulação de robótica mais famosas, existindo há mais de 25 anos para matlab e mais recentemente em 2019 tendo sido trazido para

python. As funcionalidades do simulador variam desde matemática espacial e transformações de coordenadas até simulações completas de manipuladores robóticos, com detecção de colisões e cálculo de torque. Há ainda a possibilidade de renderizar o robô, bem como a trajetória simulada, vide Figura 1. (CORKE; HAVILAND, 2021)

FIGURA 1. Renderização feita no Robotics Toolbox for Python

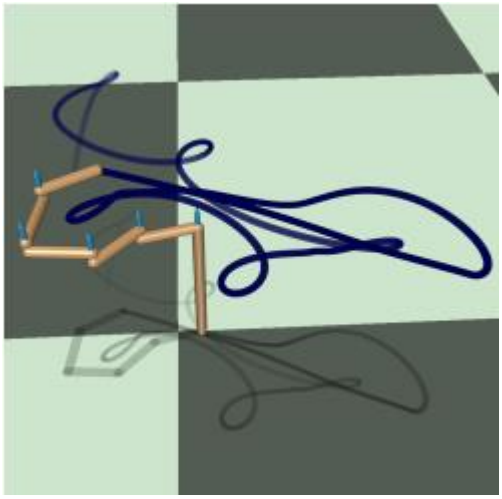


Fonte: Corke e Haviland (2021)

De outro lado o rápido desenvolvimento de aplicações de Machine Learning, em especial Deep Reinforcement Learining levou à um acelerado desenvolvimento de simuladores de física cada vez mais complexos que permitissem a aplicação desses modelos em diversos ambientes. Começando por ambientes simples em jogos de Atari, hoje é possível encontrar uma grande variedade de simuladores de física, sendo MuJoCo e Pybullet dois dos mais relevantes e usados para aplicações em robótica, especialmente em manipuladores robóticos. Outras plataformas muito usadas são Gazebo, CopeliaSim e SimGrasp.

MuJoCo é um simulador de física atualmente sob propriedade da Google Deep Mind, subsidiária da Alphabet especializada em aplicações de aprendizado de máquina, e é disponibilizado ao público gratuitamente. Construído em C++ o simulador é bastante diverso, abrangendo vários tipos diferentes de ambientes de simulação e com recursos de integração para ambientes de aprendizado de máquina muito interessantes (EREZ; TASSA; TODOROV, 2015). Entretanto, para o caso específico dos manipuladores robóticos ele acaba perdendo em algumas funcionalidades muito importantes como planejamento de trajetória e cinemática inversa. É possível ainda renderizar o robô e a trajetória simulada, conforme Figura 2. (TORODOV; ERES; TASSA, 2012)

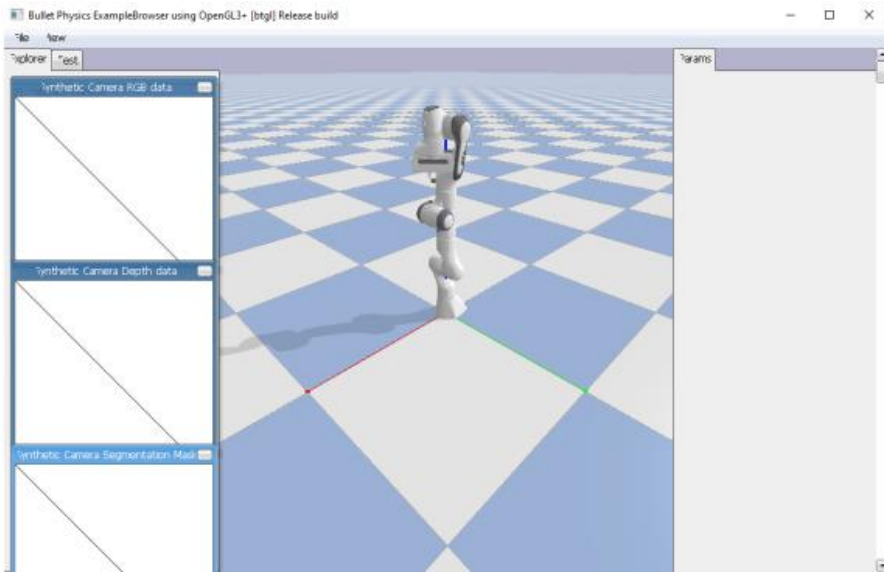
FIGURA 2. Renderização de manipulador robótico no MuJoCo



FONTE: Erez, Tassa e Todorov (2015)

Pybullet é um simulador baseado em uma já conhecida biblioteca de colisões e dinâmica de corpos rígidos, Bullet, escrita em C++. Além de bastante completo na simulação de manipuladores robóticos, o simulador ainda tem outras vantagens. Vindo de uma família de projetos open source (código aberto), Pybullet tem muitas extensões e adições construídas pela comunidade, incluindo Pybullet Gymperium, uma implementação da famosa biblioteca de ambientes de aprendizado de máquina Gym especificamente para o simulador. Ele ainda possui capacidade de renderização da trajetória simulada, conforme Figura 3. (ELENBERGER, 2018)

FIGURA 3. Renderização de manipulador no Pybullet



FONTE: Próprio Autor

3. ANÁLISE DOS SIMULADORES

Dentro das várias ferramentas disponíveis existem aquelas que se focam mais em robótica, como Robotics Toolbox e Pybotics, e aquelas mais direcionadas a aprendizado de máquina, como MuJoCo e Pybullet. Enquanto as duas primeiras entregam mais opções na construção dos robôs, maior variedade de opções de planejamento de trajetória e um acesso melhor aos detalhes das juntas e de cada etapa dos cálculos, as duas últimas entregam uma boa integração com modelos de aprendizado de máquina, processamento mais rápido e uma maior comunidade ativa por trás deles, o que se traduz em menos bugs e problemas e mais documentação.

Ao trabalhar com Deep Reinforcement Learning é comum a necessidade de realização de milhares ou dezenas de milhares de episódios (simulações) cada uma com milhares de passos. Quando o uso de estratégias evolucionárias entra em questão, vários modelos precisam rodar ao mesmo tempo competindo entre si, o que pode elevar essa demanda computacional em algumas centenas de vezes.

Nesse sentido, simuladores construídos em C++ como MuJoCo e Pybullet levam uma enorme vantagem sobre simuladores construídos em Python como Robotics Toolbox e Pybotics, que são construídos em Python que por sua vez é escrito em C. Além da camada extra na compilação do código, a comunidade maior e, no caso do MuJoCo o fato do código ser mantido e atualizado por uma empresa dedicada, resultam em mais eficiência e menos bugs, algo fundamental à velocidade do desenvolvimento.

Comparando os desafios da simulação em robótica com os desafios da construção de ambientes de aprendizado de máquina, temos que a simulação em robótica já é muito bem estabelecida e utilizada enquanto o os ambientes de aprendizado de máquina ainda passam por rápido desenvolvimento e é são enormemente complicados em variedade de parâmetros a serem otimizados e conectados. A diferença entre um simulador de física voltado para robótica para um simulador de física voltado para aprendizado de máquina em tarefas simples, como é o caso do que é proposto aqui (executar uma trajetória e calcular a energia gasta), não é significativa para os propósitos desse trabalho. Já a diferença em recursos e conectividade com pacotes de aprendizado de máquina e construção de ambientes entre os simuladores voltados para robótica e os voltados para aprendizado de máquina é muito significativa, dando uma vantagem para os últimos.

Dentre os simuladores mais focados em aprendizado de máquina, MuJoCo e Pybullet oferecem muitos recursos para construção de ambientes, em especial integração com Gym, a biblioteca mantida pela Open AI mais usada para esse tipo de tarefa. Pybullet em especial tem uma versão própria dessa biblioteca chamada Pybullet Gymperium que não só a integra com Gym, mas também já traz modelos de Deep Reinforcement Learning da biblioteca Tensorforce, do Tensorflow, já integrados para rápido desenvolvimento.

Plasencia et Al (2019) avaliou as ferramentas de construção de ambiente, mais especificamente Gym e V-REP. A conclusão obtida foi que Gym ganha na facilidade de uso, padronização e suporte da comunidade enquanto V-REP fornece funções mais elaboradas e um controle mais profundo, às custas de mais complexidade e trabalho para configurar funções básicas. Dentro do propósito do presente trabalho Gym é considerado a melhor opção.

Staranowicz e Mariottini (2011) trazem uma comparação mais voltada ao controle de robôs industriais. Nela aparecem alguns simuladores mais gerais como o Gazebo, mas a análise se foca principalmente em bibliotecas como ROS, Sinbad e CARMEN, com um viés para simuladores que funcionem com programação em mais baixo nível, oferecendo mais controle e eficiência, mas à um custo muito alto para gerenciar todas as conexões necessárias para treinar um modelo de Deep Reinforcement Learning.

Körber et Al (2022) compara Gazebo, MuJoCo, Pybuller e Webots em eficiência de uso de recursos computacionais, rodando ambos na mesma configuração de máquina em dois cenários diferentes: Um manipulador robótico pegando e posicionando cilindros em uma mesa e um cubo de 216 esferas caindo no chão causando vários contatos simultâneos. O simulador que executou mais rápido various entre os cenários com PyBullet se saindo melhor no primeiro e MuJoCo no segundo. Um sumário dos resultados obtidos e do gasto computacional são apresentados na Figura 4.

FIGURA 4. Média de consumo computacional e de correspondência de tempo real (RTF) para passo de 1ms nas execuções com cada um dos simuladores.

		Server		Mobile workstation		Notebook	
		CPU	*RTF*	CPU	*RTF*	CPU	*RTF*
Gazebo	robot	273.0 %	4.3	264.4 %	4.4	(11.9 %)	2.3
	spheres	223.1 %	1.1	208.7 %	1.4	213.2 %	0.8
MuJoCo	robot	116.3 %	2.4	100.8 %	2.8	103.1 %	2.1
	spheres	113.4 %	0.6	99.8 %	0.8	102.8 %	0.6
PyBullet	robot	119.3 %	0.7	102.0 %	0.8	104.0 %	0.6
	spheres	118.0 %	1.1	100.7 %	1.3	103.8 %	1.0
Webots	robot	124.3 %	1.8	105.6 %	1.7	109.1 %	1.3
	spheres	120.7 %	1.2	103.8 %	1.2	103.3 %	0.5

FONTE: Körber et Al (2022)

Na Figura 4 são mostrados os resultados das duas simulações "robot" e "spheres" para cada um dos simuladores analisados em duas métricas diferentes: Uso de CPU e a correspondência de tempo real de quanto tempo 1ms de simulação demora a ser processado na máquina utilizada. Comparando os resultados, PyBullet parece ser o simulador mais eficiente computacionalmente.

Collins et Al (2021) fizeram um estudo comparativo considerando várias aplicações e montaram um sumário das principais características presentes e ausentes em cada um conforme Figura 5.

FIGURA 5. Tabela comparativa das principais características de cada simulador.

Simulator	Pathplanning	Inverse Dynamics	Inverse Kinematics	Suction	Deformable Objects	Force/Torque Sensor	Realistic Rendering
SimGrasp	✓	✓	✓	✗	✗	✓	✗
Gazebo	✓	✓	✓	✗	✗	✓	✗
CoppeliaSim	✓	✗	✓	✓	✗	✓	✗
Pybullet	✓	✓	✓	✗	✓	✓	✗
MuJoCo	✗	✓	✗	✗	✓	✓	✗
NVidia Isaac	✓	✗	✓	✓	✓	✓	✓

Fonte: Collins et al. (2021)

Conforme é possível perceber a ausência de cinemática inversa aliada à ausência de ferramentas de planejamento de trajetória tornam mais penosa a utilização do MuJoCo em comparação ao Pybullet no quesito simulação de robótica.

A opção de renderização realística refere-se ao uso de engines como a da Unity para gerar visualizações bem definidas e detalhadas, como gráficos de jogos. Apesar de desejável, é um recurso supérfluo visto que os dois simuladores analisados possuem mecanismos próprios de renderização que, apesar de mais simples, atendem perfeitamente às demandas.

4. CONCLUSÕES

O emprego de manipuladores robóticos industriais nunca foi tão grande e não apresenta sinais de desaceleração. Por consequência, o desenvolvimento de pesquisa sobre eles se torna cada vez mais relevante, em especial aquela voltada à redução de consumo de energia que pode representar ganhos muito significativos tanto em receita quanto em sustentabilidade. No contexto da otimização do consumo de energia em manipuladores robóticos os modelos computacionais têm apresentado os melhores resultados e há um grande potencial para a aplicação de Deep Reinforcement Learning para o problema.

A construção do ambiente de aprendizado, porém, tem seus desafios, com dezenas de simuladores disponíveis construídos das mais variadas formas, alguns mais voltados para robótica e outros para simulação. Sendo a simulação em robótica estabelecida há mais tempo e a aplicação em questão necessitando uma baixa complexidade da mesma, é preferível utilizar simuladores mais voltados a machine learning, que apresentam mais recursos nessa área. Além disso, estes são os simuladores mais usados, e justamente por isso os mais bem mantidos e com melhor documentação.

Vários dos simuladores voltados para machine learning são pouco focados em manipuladores robóticos, fornecendo uma ampla gama de ambientes de simulação para os mais variados tipos de tarefas para aprendizado, mas não necessariamente os melhores recursos para a aplicação específica

de manipuladores. Nesse contexto, o Pybullet é superior, contando com mais recursos por isso é a escolha deste trabalho.

DECLARAÇÃO DE RESPONSABILIDADE

Os autores são os únicos responsáveis por este trabalho.

REFERÊNCIAS

CORKE, P.; HAVILAND, J. Not your grandmother's toolbox – the Robotics Toolbox reinvented for Python. 30 maio 2021.

COLLINS, J. et al. A Review of Physics Simulators for Robotic Applications. IEEE Access, v. 9, p. 51416–51431, 2021.

ELLENBERGER, B. benelot/pybullet-gym. Disponível em: <https://github.com/benelot/pybullet-gym/tree/master>. Acesso em: 7 ago. 2023.

EREZ, T.; TASSA, Y.; TODOROV, E. Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX. 2015 IEEE International Conference on Robotics and Automation (ICRA), maio 2015.

FORTUNE BUSINESS INSIGHTS. Industrial Robots Market Size, Share & COVID-19 Impact Analysis. Market Research Report, Pune, Maharashtra, v. 1, n. 1, p. 0-10, maio 2021. Disponível em: https://www.fortunebusinessinsights.com/industry-reports/industrial-robots-market-100360. Acesso em: 20 out. 2022.

INTERNATIONAL FEDERATION OF ROBOTICS. Executive Summary World Robotics 2020: industrial robots. World Robotics, Frankfurt, v. 1, n. 1, p. 13-16, 24 set. 2020. Disponível em: https://ifr.org/searchresults/f5bc56be70516cf7a4e5f37436d21882/. Acesso em: 25 out. 2022.

KÖRBER, M. et al. Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning. [s.l: s.n.]. Disponível em: <https://arxiv.org/pdf/2103.04616.pdf>. Acesso em: 8 maio. 2022.

MEIKE, D.; RIBICKIS, L. Energy efficient use of robotics in the automobile industry. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6088567/>. Acesso em: 31 maio. 2023.

NADEAU, N. Pybotics: Python Toolbox for Robotics. Journal of Open Source Software, v. 4, n. 41, p. 1738, 30 set. 2019.

PLASENCIA, A. et al. Open Source Robotic Simulators Platforms for Teaching Deep Reinforcement Learning Algorithms. Procedia Computer Science, v. 150, p. 162–170, 2019.

STARANOWICZ, A.; MARIOTTINI, G. L. A survey and comparison of commercial and open-source robotic simulator software. Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11, 2011.

TODOROV, E.; EREZ, T.; TASSA, Y. MuJoCo: A physics engine for model-based control. Disponível em: https://ieeexplore.ieee.org/abstract/document/6386109?casa_token=6rWkZ7IMtbYAAAAA:otTbVy0JGbhNVb_Dw-WfRVsccPNvQ8DVC-X_C4dNuV4avgpV8Ov2ReM1BY2WJZz405eSmBAB

VYSOCKÝ, A. et al. Reduction in Robotic Arm Energy Consumption by Particle Swarm Optimization. Applied Sciences, v. 10, n. 22, p. 8241, 1 jan. 2020.

YIN, S.; JI, W.; WANG, L. A machine learning based energy efficient trajectory planning approach for industrial robots. Procedia CIRP, v. 81, p. 429–434, 1 jan. 2019.

YIN, S.; JI, W.; WANG, L. A machine learning based energy efficient trajectory planning approach for industrial robots. Procedia CIRP, v. 81, p. 429–434, 1 jan. 2019.