Aplicação de Aprendizado por Reforço para Controle de Orientação e Posição de um Manipulador Robótico de 6 Graus de Liberdade

Felipe R. Campos [*,***] **Aline X. Fidêncio** [**] **Gustavo Pessin** [***] **Gustavo M. Freitas** [****]

[*] *Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto, MG (e-mail: felipe.campos@aluno.itv.org)*
[**] *Faculty of Electrical Engineering and Information Technology, Ruhr-University Bochum, Germany (e-mail: aline.xavierfidencio@ruhr-uni-bochum.de)*
[***] *Instituto Tecnológico Vale, Ouro Preto, MG (e-mail: gustavo.pessin@itv.org)*
[****] *Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG (e-mail: gustavomfreitas@ufmg.br)*

Abstract: Applications with autonomous robots play an important role in the industry and in everyday life. Among them, the activities of manipulating and moving objects are highlighted by the wide variety of possible applications. These activities in static and known environments can be implemented through logic planned by the developer, but this is not feasible in dynamic environments. Machine learning techniques such as Reinforcement Learning (RL) algorithms have sought to replace the pre-defined programming by teaching the robot how to act. This paper presents the implementation of two RL algorithms, Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO), for orientation and position control of a 6-degree-of-freedom (6-DoF) robotic manipulator. The results demonstrated that the DDPG had a faster learning convergence in simpler activities, but if the complexity of the problem increases, it might not obtain a satisfactory behavior. On the other hand, PPO can solve more complex problems, however, it limits the convergence rate to the best result in order to avoid learning instability.

Resumo: Aplicações com robôs autônomos tem desempenhado um papel importante na indústria e na vida cotidiana. Dentre elas, as atividades de manipulação e deslocamento de objetos se destacam pela ampla variedade de possíveis aplicações. Essas atividades em ambientes estáticos conhecidos podem ser implementadas por meio de lógicas previstas pelo desenvolvedor, porém isso pode ser inviável em ambientes dinâmicos. Técnicas de Aprendizado de Máquinas têm procurado substituir a programação pré-definida pelo processo de ensinar o robô como agir, utilizando por exemplo algoritmos de Aprendizado por Reforço (AR). Este artigo apresenta a implementação de dois algoritmos de AR, Deep Deterministic Policy Gradient (DDPG) e Proximal Policy Optimization (PPO), para controle de orientação e posição de um manipulador robótico de 6 graus de liberdade (6-DoF). Os resultados demonstram que o DDPG teve uma convergência mais rápida do aprendizado em atividades mais simples, porém se a complexidade do problema aumenta, ele pode não obter um comportamento satisfatório. Já o PPO consegue resolver problemas mais complexos, entretanto limita a taxa de convergência para o melhor resultado a fim de evitar instabilidade no aprendizado.

Keywords: Robotics; Machine Learning; Reinforcement Learning; DDPG; PPO.
Palavras-chaves: Robótica; Aprendizado de Máquinas; Aprendizado por Reforço; DDPG; PPO.

1. INTRODUÇÃO

Atualmente, a robótica desempenha um papel importante em ambientes industriais e na vida cotidiana, onde a maioria dos robôs executam tarefas de maneira autônoma. Na indústria, por exemplo, dispositivos robóticos podem ser utilizados em tarefas como soldagem, montagem, etiquetagem, etc. Já em um ambiente residencial, os dispositivos robóticos podem ser utilizados para cozinhar, limpar, vigiar, além de várias outras aplicações (Zhang and Liu, 2021). Por oferecerem inúmeras possibilidades de aplicações, os manipuladores robóticos são umas das principais escolhas dos desenvolvedores quando as atividades são de manipulação ou deslocamento de objetos (*pick and place*) (Vernon, 2014).

As atividades de *pick and place* podem ser divididas em subtarefas: controle de posição do manipulador que tem como objetivo executar trajetórias específicas para pegar (*pick*) e deixar (*place*) o objeto nas posições desejadas, e controle de orientação com o objetivo de manter a ferramenta do manipulador orientada adequadamente durante a realização da tarefa. Em alguns cenários bem estruturados, é possível implementar um sistema robótico para executar atividades com controles de orientação e posição de um manipulador. No entanto, a manipulação de objetos com alta variabilidade ou em ambientes menos estruturados torna a atividade mais complexa (Zhang and Liu, 2021).

As tarefas que necessitam de controle de orientação e posição de um manipulador robótico executadas em ambientes estruturados e com objetos precisamente colocados podem ser implementadas por meio de uma lógica previamente programável. Entretanto, um ambiente de grande complexidade e com várias interferências indeterminadas desfavorece a implementação de um algoritmo lógico pré-programado pois os dados obtidos poderão conter informações não previstas pelo desenvolvedor, impossibilitando assim a conclusão da tarefa. Essas complicações tornaram a utilização de robôs autônomos nas atividades em ambientes dinâmicos um tema relevante para pesquisas da área do Aprendizado de Máquina nos últimos anos (Lei et al., 2018).

As primeiras abordagens de estudo no desenvolvimento de robôs autônomos deixaram claro que um algoritmo baseado puramente no raciocínio e percepções do roboticista pode não ser capaz de abranger todas as circunstâncias necessárias para execução de tarefas em ambientes complexos. Com o avanço das pesquisas em Aprendizado de Máquina, foi possível desenvolver algoritmos adaptativos baseados nos sistemas humanos e que aprendem mediante a observação do ambiente por meio de tentativa e erro. Esse método de implementação é chamado de Aprendizado por Reforço (AR), modelo de Aprendizado de Máquina com técnicas que permitem um agente interagir com o ambiente de trabalho para que possa tomar decisões e executar as ações que maximizarão suas recompensas totais (Sutton and Barto, 2018). Assim sendo, o agente consegue aprender uma estratégia para solucionar problemas baseada em uma estrutura matemática (Academy, 2021).

Dessa forma, robôs podem aprender a executar tarefas em ambientes dinâmicos tomando decisões em situações não previstas, e a aplicação de algoritmos de AR é uma maneira de ensinar o robô a realizar uma tarefa sem especificar previamente como fazê-la. Sendo assim, as técnicas de AR oferecem alternativas que podem lidar com adversidades encontradas na implementação de atividades que necessitam do controle da orientação e posição de um manipulador robótico em ambientes dinâmicos (Iriondo et al., 2019).

Este artigo propõe o desenvolvimento de uma estratégia de AR para o controle de orientação e posição de um manipulador robótico de seis graus de liberdade (6-GdL) utilizado em tarefas de *pick and place*. Os algoritmos de AR escolhidos para a implementação são o *Deep Deterministic Policy Gradient* (DDPG), pelo fato de ser capaz de trabalhar com variáveis no espaço contínuo para as ações e estados do sistema (Tiong et al., 2020), e o *Proximal Policy Optimization* (PPO), por apresentar soluções estáveis e com implementações mais simples em ambientes mais complexos (Schulman et al., 2017). Para representar o ambiente de execução da tarefa, foi utilizado o programa de simulação robótica CoppeliaSim. Com as implementações, é esperado que o manipulador aprenda a executar de maneira eficiente o controle de orientação e posição de sua ferramenta, além de poder observar as características de cada um dos algoritmos implementados por meio dos resultados obtidos nos aprendizados. O acompanhamento do processo de aprendizado e a validação dos resultados são realizados por meio de testes em ambientes simulados. Como contribuições do artigo, é possível citar a implementação de dois algoritmos de AR para realizar o controle de orientação e posição de um manipulador robótico, além da comparação dos resultados obtidos por simulação.

O artigo está dividido em seis seções. Na Seção 2 são apresentados temas técnicos relacionados ao Processo Decisório de Markov como base para compreender o comportamento de um algoritmo de AR. Também na Seção 2 é descrita a estrutura do algoritmo de AR para o controle de orientação e posição de um manipulador robótico de 6-GdL. A Seção 3 cita os principais agentes de AR que podem ser utilizados em situações como a atividade proposta, além de descrever as características dos algoritmos utilizados para a implementação, o DDPG e o PPO. A metodologia utilizada nas simulações está descrita na Seção 4 , seguida da Seção 5 que apresenta os resultados obtidos nos testes executados. Por fim, na Seção 6, estão a conclusão e a proposta de trabalhos futuros.

2. APRENDIZADO POR REFORÇO PARA O CONTROLE DE ORIENTAÇÃO E POSIÇÃO DE UM MANIPULADOR ROBÓTICO

O principal objetivo de um algoritmo de AR é otimizar as interações dinâmicas de um agente em um ambiente imprevisível. Tal estrutura é frequentemente apresentada como um Processo Decisório de Markov (PDM).

O PDM consiste em uma maneira de modelar processos onde a escolha da ação em um determinado estado é probabilística. Em um PDM, é possível saber as informações dos estados atuais do processo e interferir nele periodicamente por meio de execução de ações (Pellegrini and Wainer, 2007).

O PDM é definido pelo conjunto de estados S possíveis em um ambiente, as possíveis ações A que o agente da tarefa pode executar, as probabilidades de transições de estados P, as recompensas R e a política π, conforme representado matematicamente pela Equação 1 (Otterlo and Wiering, 2012):

$$M = (S, A, P, R, \gamma, \pi). \tag{1}$$

Os algoritmos de AR são baseados em PDM com o objetivo de buscar a melhor recompensa por meio de iterações com o ambiente de uma atividade. Para o desenvolvimento do algoritmo de AR aplicado no controle de orientação e posição de um manipulador robótico, é necessário definir qual é o ambiente da atividade, quem é o agente, os espaços de estados e ações que o agente pode trabalhar, e a função de recompensa para que o agente conclua o aprendizado da maneira desejada (Sutton and Barto, 2018).

Ambiente: O ambiente é o espaço pelo qual o agente interage por meio das ações, incluindo obstáculos e interferências externas. O agente envia ao ambiente a ação que deseja tomar, de maneira que o ambiente retorna o novo estado em que o agente estará após executar a ação, além da recompensa adquirida.

Nesse trabalho, o ambiente utilizado para o aprendizado representa uma atividade com manipulador robótico de 6-GdL acoplado à uma mesa fixa. Este ambiente é apresentado por uma simulação em que a posição do objeto a ser deslocado, representado por uma esfera vermelha, é gerada aleatoriamente. Já a orientação dessa esfera está fixada. É por meio da iteração com esse ambiente que o agente consegue analisar o espaço de estados e decidir as melhores ações a serem tomadas.

Agente: O agente é quem toma as decisões de quais ações serão executadas no decorrer das iterações com o ambiente. Assim sendo, o agente corresponde à implementação da estratégia utilizada para tratar as informações coletadas do ambiente, atualizar os parâmetros utilizados para o aprendizado e convergi-los para um resultado ótimo a fim de executar a tarefa.

Espaço de Estados (S): O Espaço de Estados é o conjunto de estados possíveis em um determinado ambiente que engloba todas as informações instantâneas referentes à atividade.

O espaço de estados S do algoritmo desenvolvido para o problema de controle de orientação e posição de um manipulador robótico UR-5 é constituído por quatro vetores no espaço contínuo e uma variável binária. Esses vetores representam as posições q_i e velocidades $\dot{q}_i$ das juntas, orientação do alvo $\varphi_t = [\phi_t \quad \theta_t \quad \psi_t]^T$ dada pelos ângulos de *roll* (ϕ), *pitch* (θ) e *yaw* (ψ), e a posição do alvo $\mathbf{p_t} = [p_{t,x} \quad p_{t,y} \quad p_{t,z}]^T$ em relação aos eixos x, y e z do sistema de coordenadas inercial de referência. Portanto, o espaço de estados do ambiente da atividade pode ser representado pela Equação 2:

$$\begin{aligned} S = [&(q_1, q_2, q_3, q_4, q_5, q_6), (\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6), \\ &(\phi_t, \theta_t, \psi_t), (p_{t,x}, p_{t,y}, p_{t,z})]. \end{aligned} \tag{2}$$

Espaço de Ações (A): O Espaço de Ações é o conjunto de todas as ações possíveis que o agente pode executar no ambiente. Nas atividades de controle de orientação e posição de um manipulador robótico UR-5, o espaço de ações A do agente é composto por um vetor no espaço contínuo $\dot{q}_i$ que contém os valores para o controle da velocidade angular de cada junta i do manipulador respeitando os limites de posição, velocidade e aceleração do dispositivo. O espaço de ações está representado na Equação 3:

$$A = (\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6). \tag{3}$$

Recompensa (R): A Recompensa é o valor retornado após a execução de uma ação em um determinado estado que mensura o sucesso ou o fracasso dessa ação. Consequentemente, os valores de recompensas avaliam efetivamente a ação tomada pelo agente, sejam eles valores imediatos ou acumulados pela experiência adquirida.

O objetivo de um algoritmo de AR é conseguir o máximo de recompensas ao executar a atividade. Para isso, é necessário implementar uma função de recompensa que englobe os estados possíveis do agente. As recompensas que são prejudiciais para a conclusão da atividade, como tempo de execução da trajetória, erro de orientação e posição devem receber um peso negativo, ou seja, uma punição. Já as que auxiliam o agente a concluir a tarefa devem receber uma recompensa positiva.

Foram executadas duas atividades diferentes para validar o algoritmo implementado para o ambiente no CoppeliaSim: controle de orientação e controle de posição. Foi implementada uma recompensa para cada atividade.

Para o controle de orientação foram consideradas as orientações do efetuador $\varphi_e = [\phi_e \quad \theta_e \quad \psi_e]^T$ e do alvo φ_t para calcular o erro escalar e_o entre elas, como definido na Equação 4:

$$e_o = \sqrt{(\phi_t - \phi_e)^2 + (\theta_t - \theta_e)^2 + (\psi_t - \psi_e)^2}. \tag{4}$$

A recompensa para o algoritmo do controle de orientação do efetuador em relação ao alvo é dada pelo erro negativo da orientação. Assim, quanto mais similar à orientação do alvo, o erro e_o diminui levando a recompensa R_o convergir a zero. Além disso, foi adicionada uma recompensa positiva $r_o = +500$ caso o manipulador chegue na orientação desejada e consiga concluir a tarefa, ou uma recompensa negativa $r_o = -1000$ caso o manipulador colida com algum obstáculo. A Equação 5 demonstra a implementação da recompensa nesse sistema:

$$R_o = -e_o + r_o. \tag{5}$$

Para o controle de posição foram consideradas a posição do efetuador $\mathbf{p_e} = [p_{e,x} \quad p_{e,y} \quad p_{e,z}]^T$ e a posição do alvo $\mathbf{p_t}$ para calcular o erro escalar e_p entre elas, como demonstrado na Equação 6:

$$e_p = -\sqrt{(p_{t,x} - p_{e,x})^2 + (p_{t,y} - p_{e,xy})^2 + (p_{t,z} - p_{e,z})^2}. \tag{6}$$

A recompensa para o algoritmo do controle de posição do efetuador é dada pelo erro negativo da posição. Assim, quanto mais perto do alvo, o erro e_p diminui levando a recompensa R_p convergir a zero. Além disso, foi adicionada uma recompensa positiva $r_p = +500$ caso o manipulador chegue na posição desejada e consiga concluir a tarefa, ou uma recompensa negativa $r_p = -1000$ caso o manipulador colida com algum obstáculo. A Equação 7 demonstra a implementação da recompensa nesse sistema:

$$R_p = -e_p + r_p. \tag{7}$$

3. AGENTES DE AR APLICÁVEIS EM ESPAÇOS CONTÍNUOS

Atualmente existem alguns agentes de AR que podem ser utilizados para resolver problemas que utilizam variáveis no espaço contínuo para implementar o espaço de estados e ações. Os mais utilizados no estado da arte são (Fidêncio et al., 2021):

- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Delayed DDPG (TD3)
- Soft Actor-Critic (SAC)
- Truncated Quantile Critics (TQC)

Desses agentes, foram selecionados o DDPG e o PPO para o aprendizado do controle da orientação e posição de um manipulador robótico de 6-GdL, pois apresentaram resultados significativos em trabalhos científicos como os de Saeed et al. (2021), Franceschetti et al. (2020) e Iriondo et al. (2019) que abordam o aprendizado de atividades robóticas utilizando técnicas de AR. Para a implementação, foram considerados como referência os agentes compartilhados pela biblioteca de código aberto Stable Baselines3 (SB3) (Raffin et al., 2019).

O DDPG é uma técnica de AR que combina *Q-learning* e Gradientes de Política e é implementado com duas redes neurais: Ator e Crítico. O Ator corresponde a uma rede de políticas que toma o estado como entrada e produz a ação exata, em vez de uma distribuição de probabilidade sobre as ações. O Crítico é uma rede de valor Q que recebe o estado e a ação como entrada e emite o valor Q. Por esse motivo, o DDPG é considerado um modelo Ator-Crítico. Portanto, o DDPG é um algoritmo que aprende uma função Q usando simultaneamente dados fora da política e a equação de Bellman, e com essa função Q encontra a melhor política π (Silver et al., 2014).

O PPO é um agente de AR derivado do método Trust Region Policy Optimization (TRPO) com aprendizado baseado na variação da política. Ele é um agente de gradiente de política proposto em 2017 pela OpenAI que visa limitar a atualização da política com objetivo de manter as características do aprendizado anterior para garantir que o Ator não desvie do aprendizado (Schulman et al., 2017).

As redes neurais utilizadas para o Ator e o Crítico dos agentes DDPG e PPO foram baseadas nos algoritmos já implementados pelo Stable Baselines3. Elas possuem três camadas ocultas com 256 neurônios cada, sendo utilizado o ReLu como função de ativação para as camadas ocultas. A taxa de aprendizado utilizada para a rede neural *Actor* foi de 10^{-4}. Para a rede neural *Critic*, a taxa de aprendizado utilizada foi de 3×10^{-4}. O valor atribuído para o fator de desconto γ, que mensura o peso das experiências futuras esperadas em relação à experiência imediata, foi de $0,99$. O algoritmo otimizador escolhido para o aprendizado foi o ADAM, algoritmo baseado em gradiente de primeira ordem de funções objetivas estocásticas. Para cada época, a rede do Ator e a rede do Crítico são atualizadas 10 vezes.

No algoritmo PPO, o hiperparâmetro ϵ, que determina o limite permitido para a atualização da política, foi definido como 0,2. Os lotes de aprendizado, chamados de *minibatches*, foram dimensionados em 125. Ambos também foram definidos de acordo com o algoritmo PPO implementado pelo Stable Baselines 3.

No algoritmo DDPG foi utilizada uma memória para a repetição de experiências, chamada de *Replay Buffer*. Sua função é embaralhar dados de experiências anteriores e utilizá-los a cada passo de atualização no processo de aprendizado. Para que o algoritmo tenha um comportamento estável, essa memória tem que ter um tamanho que acumule experiências suficientes, entretanto sem sobrecarregar o sistema para não retardar o aprendizado. Depois de alguns testes, o *Replay Buffer* foi empiricamente definido em 10^6. Uma das características do DDPG é a possibilidade de abordar a exploração de novos dados independentemente do algoritmo de aprendizagem, e isso é possível por meio de uma amostragem de ruído de exploração σ incluído à política do ator. Para isso, foi definido um ruido de exploração σ no valor de 0,1.

Para os cálculos necessários para o aprendizado, foram utilizadas funções fornecidas pela biblioteca Pytorch, OpenAi Gym e PyRep. Para a plotagem dos resultados, foi utilizada a ferramenta TensorBoard, fornecida pela biblioteca TensorFlow.

4. SIMULAÇÕES

Para o estudo e avaliação dos algoritmos DDPG e PPO, o ambiente virtual utilizado nas tarefas e de controle de orientação e posição de um manipulador de 6-GdL para o estudo de caso foi implementado de acordo com os padrões do OpenAI Gym, que é um conjunto de bibliotecas de ferramentas utilizadas para desenvolver e comparar algoritmos de AR (Brockman et al., 2016). O OpenAI Gym é utilizado frequentemente para testes, pois fornece uma variedade de ambientes padronizados (*benchmarks*) para algoritmos de AR. Além disso, as especificações desses ambientes estão abertas para os desenvolvedores com intuito de possibilitar a integração de ambientes personalizados.Logo, foi possível padronizar o ambiente de simulação do *software* CoppeliaSim de acordo com os parâmetros do OpenAI Gym. Os algoritmos fornecidos pela Stable Baselines 3 podem ser utilizados para o aprendizado da atividade no ambiente de simulação do CoppeliaSim (Raffin et al., 2019).

O ambiente simulado para a implementação do algoritmo foi desenvolvido com a utilização do CoppeliaSim, ilustrado na Figura 1, é composto pelo manipulador robótico UR5 fixado em uma mesa para simular atividades em um

laboratório, e uma esfera vermelha que representa o objeto alvo a ser coletado e deslocado. Apesar de uma esfera ser um objeto uniforme cuja orientação não interfere na atividade, é importante o controle de orientação do robô durante a manipulação de objetos mais complexos.

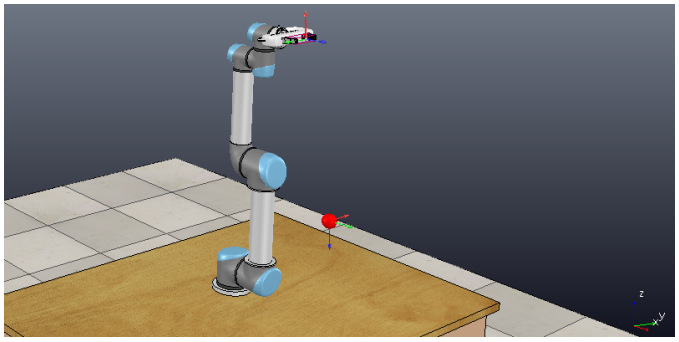


Figura 1. Ambiente de simulação no CoppeliaSim para atividades com um manipulador.

O algoritmo também utiliza da biblioteca PyRep, um conjunto de ferramentas para pesquisa de aprendizado de robôs em simulações no CoppeliaSim, a fim de modelar o ambiente da atividade. O PyRep possibilita a extração dos valores de estado encontrados no ambiente de simulação do CoppeliaSim, bem como a interação com simulação por meio das ações do algoritmo de aprendizado (James et al., 2019).

5. RESULTADOS

Os resultados de validação dos algoritmos obtidos estão apresentados mediante simulação no CoppeliaSim com a utilização dos algoritmos DDPG e PPO para o controle de orientação e posição da ferramenta do manipulador. Os valores apresentados nos gráficos são referentes aos resultados acumulados por episódio. Um vídeo ilustrando algumas das simulações controlando o manipulador robótico está disponível em `https://github.com/FelipeRigueira/UR-5_DDPG_PPO.git`

5.1 Controle de Orientação

Tanto o algoritmo DDPG quanto o PPO conseguiram convergir o aprendizado para a atividade de controle da orientação da ferramenta acoplada ao manipulador. Dessa maneira, ambos os agentes treinados conseguiram manter a execução da atividade de maneira desejada. Entretanto, o algoritmo DDPG precisou de menos tempo de aprendizado para convergir.

O aprendizado com o agente DDPG apresentou os resultados ilustrados na Figura 2a em relação à duração dos episódios no decorrer da execução do algoritmo. Foi concluído que a duração dos episódios teve uma redução muito rápida, chegando à um resultado ótimo antes mesmo dos 100 mil passos de aprendizado, conseguindo concluir a tarefa com menos de 40 passos.

Inversamente proporcional à duração, os resultados referentes às recompensas podem ser observados na Figura 2b. Houve um aumento de seus valores obtidos por cada episódio mantendo assim próximo de 500, o que garante a conclusão da tarefa de maneira adequada.

Os resultados obtidos em relação à duração dos episódios com o algoritmo PPO para a atividade de controle de orientação demonstram na Figura 2a uma convergência mais lenta para o resultado ótimo. Após a convergência com aproximadamente 1 milhão e 200 mil passos de aprendizado, o agente PPO manteve a duração dos episódios abaixo de 40 passos para a execução da atividade, assim como o agente DDPG.

O mesmo pode ser observado no comportamento das recompensas adquiridas no decorrer do aprendizado na Figura 2b, apresentando um aumento das recompensas adquiridas no decorrer do aprendizado. Apesar da convergência lenta, após 1 milhão e 200 mil passos, a recompensa adquirida por episódio se mantém estável no valor aproximado de 500, o que demonstra que o agente consegue concluir a atividade continuamente.

Após o aprendizado com ambos os algoritmos, os agentes do PPO e do DDPG conseguiram concluir a atividade de controle de orientação como ilustrado na Figura 3. A orientação da esfera corresponde à orientação desejada a ferramenta do manipulador. Mesmo que o aprendizado do algoritmo PPO tenha demorado alguns passos a mais, o resultado final foi semelhante ao comportamento do DDPG.

5.2 Controle de Posição

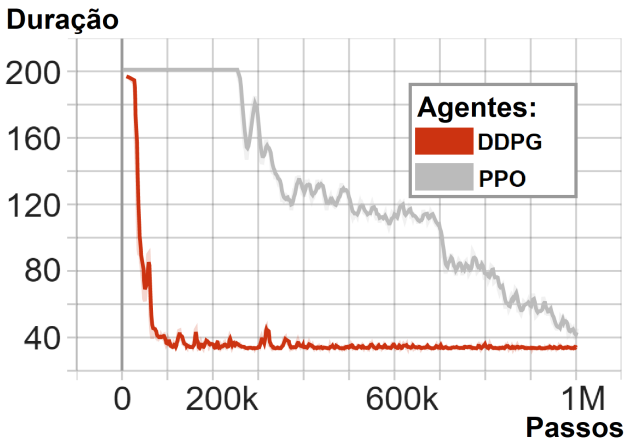
Na atividade de controle da posição do efetuador, os algoritmos tiveram comportamentos totalmente distintos no decorrer do aprendizado. Dessa vez, utilizando os parâmetros definidos na Seção 3, o Algoritmo DDPG não conseguiu ter uma convergência no aprendizado. Já o algoritmo PPO conseguiu convergir e manter a atividade sendo executada continuamente.

Em atividades mais complexas, o DDPG tem limitações no aprendizado que dificultam a convergência, principalmente por não limitar a variação dos parâmetros da rede neural que representa a política durante o aprendizado. Em vista disto, os resultados para o algoritmo na atividade de controle de posição tiveram um comportamento diferente do caso anterior.

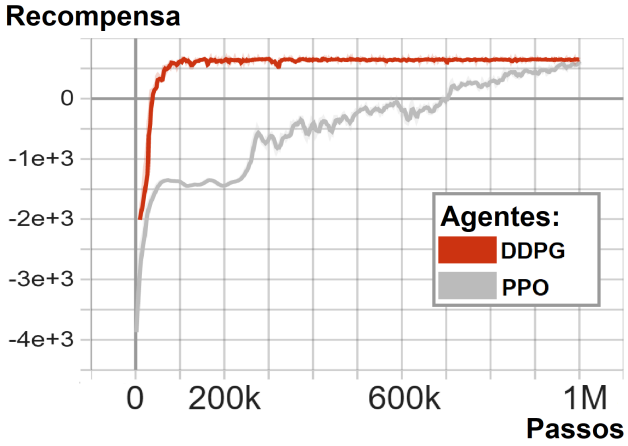
Durante o decorrer de 2 milhões de passos no aprendizado do agente, o manipulador conseguiu minimizar a distância do efetuador em relação ao alvo, porém não alcançava efetivamente o objetivo. O manipulador robótico quase concluiu a tarefa em diversos episódios, porém acabava por desviar do alvo e terminar a atividade com uma diferença de posição maior. Logo, quase todos os episódios eram interrompidos após a limitação de 200 passos por episódio ou quando havia colisão, conforme ilustrado na Figura 4a.

Como o agente não conseguiu executar a tarefa, as recompensas se mantiveram negativas durante todo o processo de aprendizado, conforme ilustrado na Figura 4b. Nessa atividade fica mais nítida a variação abrupta da política no decorrer do aprendizado, que define a estratégia do algoritmo de escolher a ação a ser executada. Por isso, o agente apresentou um comportamento instável no decorrer de 2 milhões de passos do aprendizado.

Como é possível observar na Figura 5a, o braço robótico não conseguiu alcançar o alvo com a ferramenta do mani-

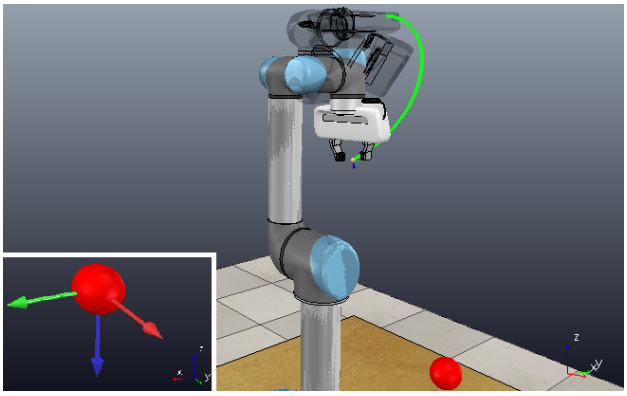


(a) Duração de cada episódio no decorrer do aprendizado da atividade.

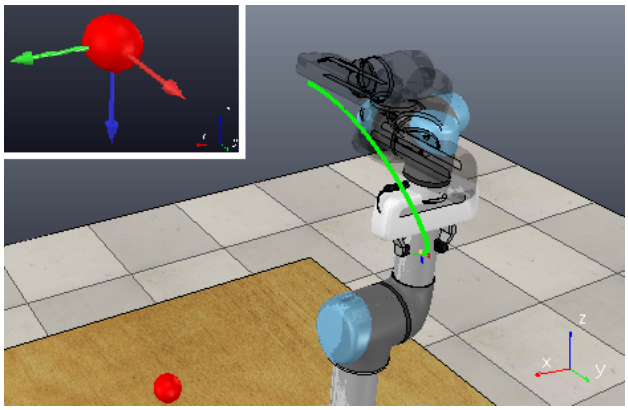


(b) Recompensa de cada episódio no decorrer do aprendizado da atividade.

Figura 2. Resultados para o controle de orientação da ferramenta utilizando os agentes DDPG e PPO.

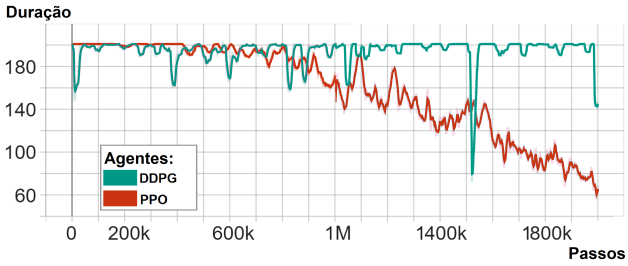


(a) Comportamento da ferramenta do manipulador após aprendizado com DDPG.

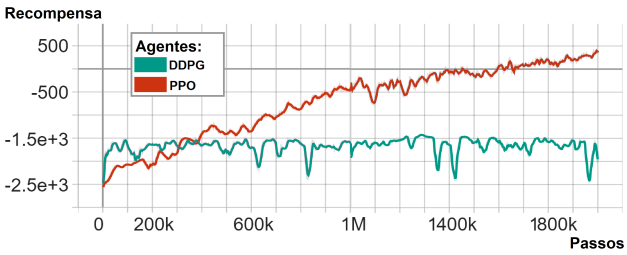


(b) Comportamento da ferramenta do manipulador após aprendizado com PPO.

Figura 3. Comportamento da ferramenta do manipulador após o aprendizado da atividade de controle de orientação com os agentes DDPG e PPO.



(a) Duração de cada episódio no decorrer do aprendizado da atividade.



(b) Recompensa de cada episódio no decorrer do aprendizado da atividade.

Figura 4. Resultados para o controle de posição da ferramenta utilizando os agente DDPG e PPO.

pulador, mesmo após 2 milhões de passos de aprendizado utilizando o algoritmo DDPG. Para um trabalho futuro, é interessante estender a quantidade de passos para observar se o agente DDPG irá convergir. Também pode ser experimentado variações no fator de desconto γ com o objetivo de tentar amenizar a instabilidade do aprendizado.

Já o PPO teve um comportamento mais estável durante o aprendizado da atividade de controle da posição, visto que o algoritmo segue uma estratégia de região de confiança baseada na limitação da variação da política de aprendizado. Isso faz com que o algoritmo demore a convergir, porém promete impedir sua instabilidade. Isso pode ser observado ao analisar a duração dos episódios na Figura 4a durante o decorrer de 2 milhões de passos do aprendizado. Após 2 milhões de passos, o agente conseguiu executar a atividade com uma média de 80 passos.

A recompensa também foi aumentando gradativamente, chegando próximo de 500 em 2 milhões de passos do

aprendizado. Após 1 milhão e 600 mil passos o agente já manteve as recompensas positivas, demonstrando que conseguia executar a tarefa, conforme ilustrado na Figura 4b.

A Figura 5b ilustra o comportamento do manipulador robótico após a conclusão do aprendizado com a utilização do PPO ao posicionar o efetuador sobre a bolinha.

6. CONCLUSÃO

Os algoritmos DDPG e PPO implementados com os parâmetros citados nesse artigo tiveram comportamentos de aprendizado distintos, sendo que cada um apresentou vantagens e desvantagens individuais, porém os dois demonstraram que podem ser utilizados e estudados em atividades robóticas. Ambos se mostraram eficientes para executar a atividade de controle de orientação, entretanto o DDPG não conseguiu manter a execução da atividade de controle de posição estável. Mesmo conseguindo executar a atividade em alguns episódios, este manteve seu comportamento instável após 2 milhões de passos de aprendizado.

O DDPG, por ser um algoritmo que não limita a atualização dos pesos da rede neural responsável pela política do agente, possibilita variações drásticas entre um episódio e outro. Devido a essa falta de limitação na atualização da política, o DDPG proporciona uma convergência muito mais rápida ao resultado ótimo, podendo encontrar um resultado de recompensa mais alto em um menor intervalo de passos que o PPO. Entretanto, o DDPG pode levar à instabilidade do agente caso essa atualização da política encontre uma recompensa que satisfaça o algoritmo naquela etapa do aprendizado, porém desviando da execução da atividade e ficando assim preso em um resultado ótimo local.

Para evitar esse tipo de instabilidade, o algoritmo PPO foi desenvolvido com parâmetros para limitar a atualização dos pesos da rede neural responsável pelo aprendizado da política, atendendo aos parâmetros que regulam uma região de confiança para essa atualização. Essa limitação na atualização da política torna o algoritmo mais lento na convergência, mas mantém o aprendizado estável durante todo o processo. Devido a essa característica, o PPO aparenta uma convergência mais lenta que o DDPG em atividades mais simples, como o controle de orientação, mas sua estabilidade no aprendizado garante a convergência em atividades mais complexas para as quais o DDPG tem mais dificuldade de apresentar resultados satisfatórios, como o controle de posição do manipulador robótico.

Além da maior estabilidade no aprendizado em relação ao DDPG, o PPO é um algoritmo que "cumpre o que promete" sendo mais prático de ser implementado, além de demandar menos processamento computacional que o DDPG. O DDPG tem parâmetros como a memória do *replay buffer* que altera significativamente o comportamento do aprendizado, podendo ser um dos motivos pelos quais ele fique instável. Visto que o PPO apresentou um comportamento estável durante o aprendizado e execução das atividades essenciais para uma tarefa de *pick and place*, ele pode ser uma boa solução para ser utilizado em trabalhos futuros.

Como trabalhos futuros é sugerida a utilização do algoritmo de PPO para aprender a executar uma atividade de controle de orientação e posição na mesma tarefa, o que pode ser feito unificando as recompensas da atividade de controle de orientação e de controle de posição em uma única função de recompensa. Assim que a atividade de controle de orientação e posição estiver concluída, seria interessante implementar uma atividade de *pick and place* adicionando uma recompensa quando o agente conseguir pegar o objeto adicionado de uma recompensa que represente a distância entre a posição atual e a desejada. Dessa maneira, é esperado que o agente aprenda a executar uma atividade completa de *pick and place*.

Outro trabalho futuro sugerido é a utilização do algoritmo implementado para a atividade de *pick and place* em uma atividade industrial, como existe no setor de mineração a manutenção das barras de um carro grelha que é utilizado no processo de pelotização do minério.

AGRADECIMENTOS

Esse artigo foi realizado com apoio da coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), da Fundação de Amparo à Pesquisa do Estado de Minas Gerais - Brasil (FAPEMIG), do Instituto Tecnológico Vale (ITV) e da Universidade Federal de Ouro Preto (UFOP) e da Universidade Federal de Minas Gerais (UFMG).

REFERÊNCIAS

Academy, D.S. (2021). Deep Learning Book. URL `https://www.deeplearningbook.com.br/`. Acesso em: 22 de junho de 2021.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Fidêncio, A.X., Glasmachers, T., and Naro, D. (2021). Application of Reinforcement Learning to a Mining System. In *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 000111–000118. IEEE.

Franceschetti, A., Tosello, E., Castaman, N., and Ghidoni, S. (2020). Robotic arm control and task training through deep reinforcement learning. *arXiv preprint arXiv:2005.02632*.

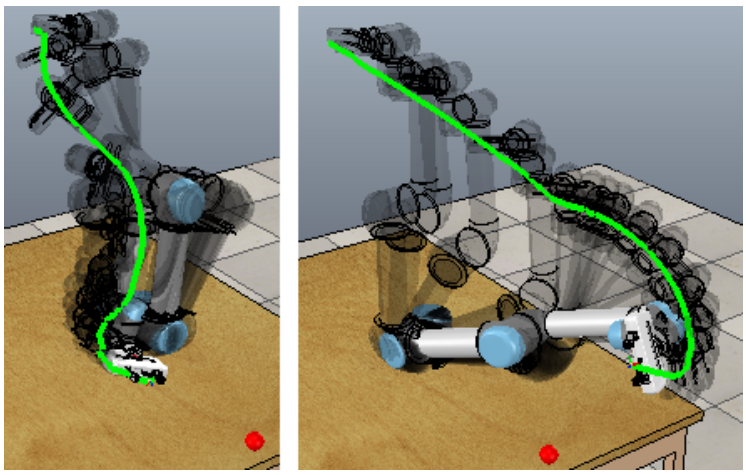
Iriondo, A., Lazkano, E., Susperregi, L., Urain, J., Fernandez, A., and Molina, J. (2019). Pick and Place Operations in Logistics Using a Mobile Manipulator Controlled with Deep Reinforcement Learning. *Applied Sciences*, 9(2), 348.

James, S., Freese, M., and Davison, A.J. (2019). Pyrep: Bringing V-REP to deep robot learning. *CoRR*, abs/1906.11176.

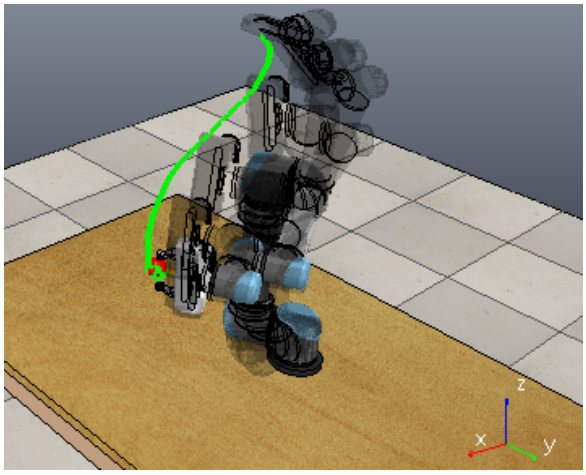
Lei, X., Zhang, Z., and Dong, P. (2018). Dynamic path planning of unknown environment based on deep reinforcement learning. *Journal of Robotics*, 2018.

Otterlo, M.v. and Wiering, M. (2012). *Reinforcement learning and markov decision processes*. Springer.

Pellegrini, J. and Wainer, J. (2007). Processos de Decisão de Markov: um tutorial. *Revista de Informática Teórica e Aplicada*, 14(2), 133–179.



(a) Comportamento instável do manipulador na atividade de controle de posição após a conclusão de 2 milhões de passos de aprendizado com o agente DDPG.



(b) Comportamento do manipulador na atividade de controle de posição após a conclusão de 2 milhões de passos de aprendizado com o agente PPO.

Figura 5. Comportamento do manipulador após aprendizado da atividade de controle de posição utilizando os agentes DDPG e PPO.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. (2019). Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`.

Saeed, M., Nagdi, M., Rosman, B., and Ali, H.H.S.M. (2021). Deep Reinforcement Learning for Robotic Hand Manipulation. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 1–5.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395. PMLR.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tiong, T., Saad, I., Teo, K.T.K., and Lago, H.b. (2020). Deep Reinforcement Learning with Robust Deep Deterministic Policy Gradient. In *2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, 1–5.

Vernon, D. (2014). *Artificial Cognitive Systems: A Primer*. MIT Press.

Zhang, B. and Liu, P. (2021). Control and benchmarking of a 7-DOF robotic arm using Gazebo and ROS. *PeerJ Computer Science*.