UNIVERSIDADE FEDERAL DE VIÇOSA

Rastreamento de trajetória para robôs agrícolas via aprendizado por reforço em ambientes simulados com modelos de incertezas

Neuller Alves Pereira
Magister Scientiae

VIÇOSA - MINAS GERAIS
2025

NEULLER ALVES PEREIRA

Rastreamento de trajetória para robôs agrícolas via aprendizado por reforço em ambientes simulados com modelos de incertezas

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Engenharia Agrícola, para obtenção do título de *Magister Scientiae*.

Orientador: Domingos S. M. Valente

Coorientadores: Daniel M. de Queiroz
Andre L. de F. Coelho

VIÇOSA - MINAS GERAIS
2025

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
P436r
2025

Pereira, Neuller Alves, 1999-
Rastreamento de trajetória para robôs agrícolas via aprendizado por reforço em ambientes simulados com modelos de incertezas / Neuller Alves Pereira. – Viçosa, MG, 2025.
1 dissertação eletrônica (53 f.): il. (algumas color.).

Orientador: Domingos Sárvio Magalhães Valente.
Dissertação (mestrado) - Universidade Federal de Viçosa, Departamento de Engenharia Agrícola, 2025.
Referências bibliográficas: f. 50-53.
DOI: https://doi.org/10.47328/ufvbbt.2026.023
Modo de acesso: World Wide Web.

1. Robótica. 2. Inteligência artificial. 3. Máquinas agrícolas - Simulação por computador. 4. Detectores. I. Valente, Domingos Sárvio Magalhães, 1978-. II. Universidade Federal de Viçosa. Departamento de Engenharia Agrícola. Programa de Pós-Graduação em Engenharia Agrícola. III. Título.

CDD 22. ed. 629.892633

Bibliotecário(a) responsável: Euzébio Luiz Pinto CRB-6/3317

NEULLER ALVES PEREIRA

Rastreamento de trajetória para robôs agrícolas via aprendizado por reforço em ambientes simulados com modelos de incertezas

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Engenharia Agrícola, para obtenção do título de *Magister Scientiae*.

APROVADA: 27 de agosto de 2025.

Assentimento:

_____

Neuller Alves Pereira
Autor

_____

Domingos Sarvio Magalhaes Valente
Orientador

Dedico à minha avó, Adeniza, que faleceu antes que esta dissertação fosse concluída. Seu amor, seu respeito e seu caráter permanecem vivos na memória de todos nós.

AGRADECIMENTOS

A Deus, que, por sua infinita misericórdia, tem me capacitado ao longo de todo o processo.
À minha família, por todo o apoio e suporte.
Ao meu orientador, professor Domingos Sárvio, pela paciência ao longo do processo de produção desta dissertação.
À Universidade Federal de Viçosa e ao Departamento de Engenharia Agrícola, pela oportunidade de realização do curso de mestrado.
Este trabalho foi realizado com o apoio das seguintes agências de pesquisa brasileiras: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Porque agora vemos por espelho em enigma, mas então veremos face a face; agora conheço em parte, mas então conhecerei como também sou conhecido.
(I Coríntios 13:12)

RESUMO

PEREIRA, Neuller Alves, M.Sc., Universidade Federal de Viçosa, agosto de 2025. **Rastreamento de trajetória para robôs agrícolas via aprendizado por reforço em ambientes simulados com modelos de incertezas**. Orientador: Domingos Sarvio Magalhaes Valente. Coorientadores: Daniel Marcal de Queiroz e Andre Luiz de Freitas Coelho.

A adoção de tecnologias de robótica na agricultura tem se intensificado nas últimas décadas, impulsionada por avanços em navegação de alta precisão, automação de tarefas e integração de sensores inteligentes. Plataformas robóticas têm se mostrado promissoras em diferentes atividades, como aplicação de defensivos agrícolas, controle de plantas daninhas, colheita, plantio, monitoramento de pragas e doenças, e automação de processos de fenotipagem. A consolidação dessas tecnologias depende de sistemas de orientação robustos, capazes de operar de forma confiável diante de falhas sensoriais, variações de relevo e irregularidades no cultivo. No contexto de deslocamento robótico, o problema de seguimento de trajetória (path tracking) consiste em guiar uma plataforma robótica ao longo de uma rota previamente definida, minimizando desvios laterais e angulares. Métodos tradicionais de controle, como Pure Pursuit, Stanley, MPC e PID, ainda são amplamente empregados, mas podem apresentar fragilidades em ambientes sujeitos a incertezas. Em contrapartida, algoritmos de Aprendizado por Reforço (Reinforcement Learning – RL) vêm surgindo como alternativas promissoras, por permitirem que agentes aprendam políticas de navegação adaptativas. Entre os algoritmos de aprendizado por reforço utilizados, destaca-se o Double Deep Q-Network (DDQN), que vêm sendo aplicado com sucesso nos últimos anos em diferentes tarefas relacionadas ao deslocamento robótico. Apesar de avanços reportados no emprego do DDQN no problema de path tracking, a literatura carece de avaliações sistemáticas e quantitativas sobre a capacidade desses modelos de aprendizado por reforço lidarem com falhas sensoriais críticas. Paralelamente, a aplicação de RL na robótica agrícola enfrenta um desafio metodológico relevante: a escassez de ambientes de simulação padronizados, realistas e reutilizáveis. Diante desse contexto, este estudo teve dois objetivos principais: (i) propor um framework de simulação paramétrico, desenvolvido em ROS2 e Gazebo, capaz de gerar áreas agrícolas virtuais com diferentes características e modelar incertezas de sensores e atuadores; e (ii) realizar uma avaliação comparativa entre o controlador Pure Pursuit e um agente DDQN no problema de path tracking, considerando tanto condições de navegação precisa quanto falhas bruscas no sinal GNSS. Para isso, uma plataforma robótica com rotação diferencial

foi modelada e equipada com sensores GNSS, LiDAR e IMU no sistema ROS2/Gazebo. Os experimentos abrangeram três áreas distintas geradas com o framework desenvolvido, totalizando 12 combinações experimentais. As áreas geradas demonstraram conformidade com os parâmetros configurados, tanto em termos de dimensões geométricas quanto na inserção dos ruídos controlados. Os resultados evidenciam que o framework proposto se apresentou como uma solução eficiente, ágil e flexível para a criação de ambientes virtuais destinados ao treinamento e à validação de modelos de controle autônomo em sistemas robóticos agrícolas. Nas áreas avaliadas, sob condições nominais (navegação de alta precisão), ambos os controladores concluíram todas as trajetórias de forma precisa, sem colisões. Sob falhas críticas do GNSS, o DDQN obteve valores inferiores de erro absoluto máximo em todas as áreas, destacando-se na prevenção de desvios extremos, obtendo um maior percentual das áreas deslocadas sem a presença de colisões, além de apresentar menor variabilidade e maior estabilidade em comparação ao Pure Pursuit. Os resultados indicam que o DDQN é capaz de desenvolver políticas de controle mais adaptativas e resilientes frente a incertezas severas, em contraste com a resposta mais rígida e reativa do controlador geométrico.

Palavras-chave: robótica agrícola; algoritmo de navegação; falhas sensoriais; inteligência artificial; máquinas agrícolas

ABSTRACT

PEREIRA, Neuller Alves, M.Sc., Universidade Federal de Viçosa, August, 2025. **Trajectory Tracking for Agricultural Robots via Reinforcement Learning in Simulated Environments with Uncertainty Models**. Adviser: Domingos Sarvio Magalhaes Valente. Co-advisers: Daniel Marcal de Queiroz and Andre Luiz de Freitas Coelho.

The adoption of robotic technologies in agriculture has intensified in recent decades, driven by advances in high-precision navigation, task automation, and the integration of intelligent sensors. Robotic platforms have shown great potential in a wide range of agricultural applications, including pesticide spraying, weed control, harvesting, planting, pest and disease monitoring, and the automation of phenotyping processes. The consolidation of these technologies relies on robust guidance systems capable of operating reliably under sensor failures, terrain variations, and crop irregularities. In the context of robotic navigation, the path tracking problem consists of guiding a robotic platform along a predefined route while minimizing lateral and angular deviations. Traditional control methods such as Pure Pursuit, Stanley, Model Predictive Control (MPC), and Proportional-Integral-Derivative (PID) controllers remain widely used, but may exhibit limitations in uncertain and dynamic environments. In contrast, Reinforcement Learning (RL) algorithms have emerged as promising alternatives, enabling agents to learn adaptive navigation policies through interaction with the environment. Among RL approaches, the Double Deep Q-Network (DDQN) has gained attention in recent years due to its successful application in various robotic navigation tasks. Despite these advances, systematic and quantitative evaluations of RL models under critical sensor failures remain scarce in the literature. Moreover, the application of RL in agricultural robotics faces a methodological challenge stemming from the lack of standardized, realistic, and reusable simulation environments. In this context, the present study had two main objectives: (i) to propose a parametric simulation framework, developed in ROS2 and Gazebo, capable of generating virtual agricultural fields with varying characteristics and modeling sensor and actuator uncertainties; and (ii) to conduct a comparative evaluation between the Pure Pursuit controller and a DDQN agent in the path tracking problem, considering both nominal navigation conditions and abrupt GNSS signal failures. To this end, a differential-drive robotic platform was modeled and equipped with GNSS, LiDAR, and IMU sensors within the ROS2/Gazebo environment. The experiments involved three distinct virtual fields generated using the proposed framework, totaling 12 experimental combinations. The generated fields exhibited

compliance with the configured parameters, both in terms of geometric dimensions and in the insertion of controlled noise. The results demonstrate that the proposed framework constitutes an efficient, agile, and flexible solution for the creation of virtual environments aimed at training and validating autonomous control models in agricultural robotics. Under nominal conditions (high-precision navigation), both controllers successfully completed all trajectories without collisions. However, under critical GNSS failures, the DDQN consistently achieved lower maximum absolute error values across all fields, outperforming Pure Pursuit in preventing extreme deviations. Furthermore, the DDQN achieved a higher percentage of collision-free coverage, while also exhibiting lower variability and greater stability compared to the Pure Pursuit controller. These findings indicate that the DDQN is capable of developing more adaptive and resilient control policies under severe uncertainties, in contrast to the more rigid and reactive behavior of the geometric controller.

Keywords: agricultural robotics; navigation algorithm; sensor failures; artificial intelligence; agricultural machinery

Sumário

1. Introdução

A adoção de tecnologias de robótica na agricultura tem se intensificado nas últimas décadas, apoiada por avanços em navegação precisa, automação de tarefas operacionais e integração de sensores a bordo. A utilização de plataformas robóticas é promissora em diferentes atividades agrícolas. Dentre elas, pode-se citar a aplicação de defensivos agrícolas (Calvert *et al*., 2021; Jin *et al*., 2023; Loukatos *et al*., 2021), erradicação de plantas invasoras (Pradel e Fays, 2022; Midtiby *et al.*, 2016), colheita (Arad *et al*., 2020; Kuznetsova *et al*., 2020), plantio (Yang *et al*., 2023; Vahdanjoo *et al*., 2023), detecção de pragas e doenças (Cubero *et al*., 2020; Fernando *et al*., 2020) e automação de fenotipagem agrícola (Atefi et al., 2021; Arunachalam e Andreasson, 2021; Xu e Li, 2022). Esses estudos colocaram em evidência distintas perspectivas das plataformas robóticas associadas com ferramentas computacionais em sistemas agrícolas, e os resultados foram promissores. De acordo com Fountas *et al.* (2020), a robótica possibilita a redução de custos nas operações, a melhoria da eficiência das operações com redução de consumo de energia, a utilização de energia limpa e conservação do solo e da água, com redução de erosão e compactação dos solos. Além disso, as aplicações de robótica oferecem possibilidades para a solução de problemas relacionados com a escassez de mão-de-obra na agricultura e substituição do trabalhador em atividades árduas (Vougioukas, 2019; Gongal *et al.*, 2015).

Um dos desafios existentes na utilização de robótica em operações agrícolas é o desenvolvimento de sistemas eficientes de orientação de deslocamento robótico. Tais sistemas devem ser capazes de lidar com fatores como falhas em sensores, variações de relevo, diferentes tipos de solo e densidade de plantio, que podem sofrer grandes variações de acordo com o local, tipo de manejo e de cultura implantada. Segundo Zhang *et al*. (2021), os sistemas de orientação mais utilizados atuam com base na identificação de linhas e fileiras de cultivo, e apresentam limitações para atuar em situações em que se têm muitos obstáculos e/ou plantas com crescimento irregular.

Alguns trabalhos recentes têm proposto sistemas de orientação para o deslocamento de robôs agrícolas. Zhang *et al*. (2021) utilizaram técnicas de visão binocular e Internet das Coisas para desenvolver um sistema de navegação capaz de se adaptar a variações nas linhas de cultivo. Yang *et al*. (2022a) desenvolveram um

algoritmo que tenta reproduzir o comportamento de um motorista para tentar reduzir os erros laterais e de direção no processo de locomoção de máquinas agrícolas autônomas. De forma geral, esses sistemas apresentaram melhoras significativas em comparação aos métodos tradicionais, evidenciando o potencial da tecnologia na orientação robótica agrícola.

No contexto de deslocamento robótico, o problema de seguimento de trajetória *(path tracking*) é fundamental, sendo amplamente explorado em aplicações de navegação autônoma. Esse problema consiste em conduzir um agente móvel ao longo de uma trajetória de referência previamente definida, minimizando, a cada instante, tanto o erro lateral — distância perpendicular entre o agente e a curva de referência — quanto o erro angular — diferença de orientação entre o robô e o trajeto. Para essa finalidade, técnicas clássicas têm sido amplamente empregadas, incluindo controladores baseados em geometria, como o *Pure Pursuit* e o *Stanley Controller*; abordagens de controle ótimo, como o Controle Preditivo Baseado em Modelo (MPC); e métodos da teoria de controle, como controladores PID e Reguladores Lineares Quadráticos (LQR) (Yao *et al*., 2020).

Mais recente (Zhang *et al.,* 2025; Sun *et al*., 2024; Yang *et al.,* 2022b), a técnica de aprendizagem por reforço (*Reinforcement Learning*) tem sido aplicada no deslocamento de robôs agrícolas. O aprendizado por reforço é um campo da inteligência artificial onde se tem a otimização de um agente para tomar decisões sequenciais, permitindo a solução de problemas complexos por meio da interação deste agente com um ambiente (Wang e Hong, 2020). Nesta classe de algoritmos, tem-se a presença de *feedbacks* na forma de recompensas ou penalidades, sendo que o agente se comporta de forma a maximizar as recompensas ao longo do tempo.

Dentre os algoritmos que implementam técnicas de *Reinforcement Learning* (RL), destaca-se o *Double Deep Q-Network* (DDQN). Essa abordagem, proposta por Van Hasselt et al. (2016), aprimora *o Deep Q-Network* (Mnih *et al*., 2015) ao mitigar a superestimação de valores de ação por meio da separação entre as etapas de seleção e avaliação das ações. Diversos estudos têm aplicado com sucesso a arquitetura DDQN para resolver problemas no contexto de atividades agrícolas (Yue *et al*., 2023; Alibabaei *et al*., 2022; Wenyu *et al*., 2019).

Apesar de avanços reportados no emprego do DDQN no problema de path tracking, a literatura carece de avaliações sistemáticas e quantitativas sobre a capacidade desses modelos de aprendizado por reforço lidarem com falhas sensoriais

críticas. Paralelamente, a aplicação de RL na robótica agrícola enfrenta um desafio metodológico relevante: a escassez de ambientes de simulação padronizados, realistas e reutilizáveis. Com frequência, pesquisas recorrem a simuladores desenvolvidos de forma *ad hoc*, o que compromete a reprodutibilidade e inviabiliza comparações justas entre diferentes políticas de controle. Tal cenário reforça a necessidade de um *framework* aberto e paramétrico para servir como base comum para o avanço da área.

Nesse contexto, o presente trabalho tem os seguintes objetivos: (1) o desenvolvimento de um *framework* de simulação customizável, implementado no Robot Operating System 2 (ROS2), apto a gerar cenários agrícolas paramétricos e a modelar incertezas sensoriais e de atuação; e (2) a condução de uma análise comparativa, utilizando tal *framework*, para investigar os *trade-offs* entre precisão e robustez no desempenho do controlador *Pure Pursuit* e de um agente DDQN sob falhas bruscas do sensor GNSS.

2. Processos de decisão de Markov e aprendizado por reforço profundo

Algoritmos de aprendizado por reforço são empregados para solucionar problemas baseados em processos de decisão de Markov (MDP), que são modelos utilizados para tomada de decisão em ambientes nos quais as transições entre estados podem ser modeladas de forma probabilística.

Em um MDP, um determinado agente interage com um ambiente, e, a cada instante, encontra-se em um estado (**s**). O sistema satisfaz a propriedade de Markov se, ao executar uma determinada ação (**a**), o agente migrará para um próximo estado (**s'**), não sendo necessárias informações do passado para definir este estado (Sutton e Barto, 2018). Como a ocorrência de eventos é estocástica, o estado **s'** não é conhecido, mas é possível descrever a probabilidade de o agente assumir cada um dos estados pertencentes ao conjunto de estados possíveis (**S**) (Sutton e Barto, 2018). Além disso, em um MDP define-se uma função de recompensa associada à execução da ação **a** no estado **s** e à transição para o estado **s'**, para todos os pares **s**, **s'** $\in$ **S**.

O processo de solução de um MDP consiste em tomar decisões sequenciais ao longo da interação do agente com o ambiente. Dá-se o nome de Política (**π**) à função que realiza a escolha da ação **a** pertencente ao conjunto de ações possíveis (**A**) que será tomada pelo agente em cada estado **s** $\in$ **S** ao longo da interação (Sutton e Barto, 2018). A soma das recompensas individuais obtidas em todas as ações adotadas pelo agente, de um estado inicial até o final da interação, é denominada Retorno (**G**) da Política (Chen *et al*., 2023; Sutton e Barto, 2018). Devido às características estocásticas dos MDPs, duas interações diferentes do agente com um mesmo sistema, em um mesmo estado, seguindo uma mesma política **π**, pode levar a diferentes estados, e, consequentemente, a diferentes valores de **G** (Sutton e Barto, 2018). Dada esta característica, a função Q(s,a) representa a esperança do Retorno obtido ao iniciar no par (**s**,**a**) e seguir a política **π** até o fim da interação. Em um processo de transição de estados, este valor pode ser calculado utilizando o conceito de esperança matemática, conforme apresentado na Equação 1.

$$Q(s,a) = \sum P(s,a,s')[R(s,a,s') + \gamma V(s')] \quad (1)$$

Em que,

$Q(s,a)$ = Esperança do Retorno.

$P(s, a, s')$ = Probabilidade de transição de um estado **s** para o estado **s'**, dado que uma ação **a** foi tomada.

$R(s, a, s')$ = Recompensa obtida ao se tomar a ação **a** no estado **s** e migrar para o estado **s'**.

γ = Fator de desconto.

$V(s')$ = Valor esperado do Retorno acumulado a partir do estado **s'**.

O Fator de desconto (**γ**) é um parâmetro adimensional, que varia de zero a um e permite regular o impacto das recompensas futuras sobre o valor do Retorno (Sutton e Barto, 2018). Dada uma política determinística, a ação escolhida em um mesmo estado em momentos diferentes será sempre a mesma. Sendo assim, o Retorno esperado da política em um estado **s** será igual ao Retorno esperado da ação adotada pela política nesse estado, conforme apresentado na Equação 2:

$$V_{\pi}(s) = Q_{\pi}(s, \pi(s)) \tag{2}$$

Em que,

$V_{\pi}(s)$ = Retorno esperado de uma política **π** dado que o agente se encontra no estado **s**.

Em um MDP, tem-se por objetivo maximizar a recompensa obtida pelo agente ao longo da execução das ações, ou seja, alcançar máximo Retorno possível (Sutton e Barto, 2018). Dado que uma determinada ação foi escolhida, o máximo Retorno esperado pode ser calculado de acordo com o definido na Equação 3.

$$Q_{opt}(s, a) = \sum P(s, a, s')\left[R(s, a, s') + \gamma V_{opt}(s')\right] \tag{3}$$

Em que,

$Q_{opt}(s, a)$ = Retorno esperado ao tomar a ação **a** no estado **s** assumindo que o agente seguirá a política ótima em todas as decisões subsequentes.

$V_{opt}(s')$ = máximo Retorno esperado a partir do estado sucessor **s'**, assumindo que, a partir desse ponto, o agente fará apenas escolhas ótimas.

O valor máximo esperado para o Retorno em um estado **s**, pode então ser obtido tomando o maior valor de Q_{opt},conforme apresentado na Equação 4.

$$V_{opt}(s) = max_a \ Q_{opt}(s,a) \quad (4)$$

Em que,

$V_{opt}(s)$ = Valor ótimo para o Retorno, dado que o agente se encontra no estado **s.**

$max_a \ Q_{opt}(s,a)$ = Valor máximo dentre os valores de Q_{opt} das ações possíveis.

Sendo assim, a política ótima em um determinado estado, é aquela que leva à escolha da ação que resulta no maior valor de Q_{opt} (Equação 5).

$$\pi_{opt}(s) = argmax_a \ Q_{opt}(s,a) \quad (5)$$

De uma forma geral, em problemas complexos, as Equações 2 a 5 podem ter soluções aproximadas com a utilização de algoritmos iterativos, de forma que haja convergência para a política ótima (Sutton e Barto, 2018; Chen *et al*., 2023). Em situações nas quais as probabilidades de transição não são conhecidas, algoritmos de aprendizado por reforço, como o Q-Learning, tentam encontrar a política ótima fazendo a exploração do ambiente, por meio de múltiplas simulações da interação, com o objetivo de estimar Q_{opt} diretamente (Sutton e Barto, 2018).

No Q-Learning, os valores de Retorno são armazenados na chamada “Tabela Q” (Figura 1), que consiste em uma tabela na qual as colunas são as possíveis ações que o agente pode executar, e as linhas os estados que ele pode assumir (Sutton e Barto, 2018).

	AÇÕES			
	a1	**a2**	...	**an**
s1	Q(s1,a1)	Q(s1,a2)	...	Q(s1,an)
s2	Q(s2,a1)	Q(s2,a2)	...	Q(s2,an)
⋮	⋮	⋮	⋱	⋮
sn	Q(sn,a1)	Q(sn,a2)	...	Q(sn,an)

ESTADOS

Figura 1 - Representação da Tabela Q. Nas colunas estão presentes as possíveis ações que o agente pode executar. Nas linhas estão os estados que ele pode assumir.

A tabela é preenchida com o Retorno esperado para cada par estado/ação, e esse valor é atualizado à medida que são realizadas simulações de interação do agente com o ambiente (Sutton e Barto, 2018). A tabela é iniciada com valores aleatórios (geralmente zero), e é atualizada com base na recompensa obtida na ação executada no estado atual, e o valor máximo de Retorno observado nas simulações anteriores para o estado alcançado após executar esta ação (Sutton e Barto, 2018), conforme apresentado na Equação 6.

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha\{[R(s,a,s') + \gamma V_{opt}(s')] - Q_{t-1}(s,a)\} \quad (6)$$

Em que,

$Q_t(s,a)$ = valor do Retorno esperado, dado que uma ação **a** foi tomada na iteração atual.

$Q_{t-1}(s,a)$ = valor do Retorno esperado, dado que uma ação **a** foi tomada na iteração anterior.

α = Taxa de aprendizado do algoritmo.

Substituindo a Equação 4 na Equação 6, torna-se possível executar as iterações com base nos valores armazenados na tabela Q (Equação 7).

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha\{[R(s,a,s') + \gamma max_a, Q_{opt}(s',a')] - Q_{t-1}(s,a)\} \quad (7)$$

A taxa de aprendizado (**α**) é um número que varia de 0 a 1 e representa o impacto que o Retorno obtido na iteração atual terá sobre o valor que será atualizado. Nas primeiras simulações, quando a tabela Q é iniciada com valores aleatórios, geralmente tem-se a utilização de **α** próximo ou igual a 1. À medida que o agente aprende, geralmente tem-se uma diminuição do valor de **α** (Sutton e Barto, 2018).

Na grande maioria das vezes, os ambientes do mundo real assumem uma quantidade muito grande de estados, de forma que a elaboração e atualização da tabela Q se torna computacionalmente inviável (Mnih *et al*., 2015). Nesses casos, uma possível solução está na substituição do uso da tabela Q por aproximação de função. Para isso, utiliza-se uma função que, com base nos dados do estado em que o agente se encontra, obtém um valor de Retorno para cada ação que pode ser tomada. Em

outras palavras, o valor de $Q_{opt}(s,a)$ é aproximado por uma função $Q(s,a,w)$, sendo que a matriz de pesos w é obtida ao longo da interação do agente com o ambiente (Mnih *et al*., 2015). Dessa forma, a cada estado, a ação mais vantajosa pode ser obtida de acordo com a Equação 1. Na arquitetura Deep Q Network (DQN) (Mnih *et al*., 2015) tem-se a utilização de uma rede neural como função de aproximação. Nesse caso, a rede neural é treinada por meio da minimização de uma função de perda cujos alvos são atualizados a cada iteração. Para promover a estabilidade do treinamento, o DQN utiliza uma rede alvo, cujos pesos são mantidos fixos por um determinado número de iterações, sendo atualizados periodicamente a partir da rede principal (Mnih *et al*., 2015), conforme apresentado na Equação 8.

$$L_i(w_i) = \{[R(s,a,s') + \gamma max_{a'} Q(s',a';w^-)\,] - Q(s,a;w_i)\}^2 \quad (8)$$

Em que,

$L_i(w_i)$ = função de perda na iteração atual.

$max_{a'} Q(s',a';w^-)$ = máximo Retorno esperado no próximo estado **s'**, estimado com os pesos da rede neural alvo.

$Q(s,a;w_i)$ = previsão atual do valor do Retorno para a ação **a** no estado **s**.

Um problema recorrente nas arquiteturas derivadas do Q-Learning é a superestimação de valores de Retorno (Thrun e Schwartz, 1993). Van Hasselt *et al.* (2016) mostraram que a superestimação do valor do Retorno que ocorre ao longo do processo de convergência da arquitetura DQN pode afetar significativamente o desempenho do modelo, podendo comprometer a convergência e levar a políticas subótimas. Estes autores propuseram então a arquitetura Double DQN, que tem por objetivo suavizar a superestimação de valores de Retorno ao longo do treinamento do modelo.

Para reduzir a superestimação do valor do Retorno, no Double DQN, a ação a ser tomada pelo agente e avaliação desta ação são realizados separadamente (Van Hasselt *et al*., 2016). Conforme acontece no DQN, a cada iteração utiliza-se uma rede neural para estimar os valores de Retorno de cada ação, permitindo a seleção daquela mais vantajosa. Entretanto, diferentemente do DQN convencional, no qual a mesma rede é utilizada tanto para estimar quanto para selecionar o valor máximo de Retorno,

no Double DQN a seleção da ação e a avaliação do seu valor são realizadas por redes distintas. Embora o DQN também utilize uma rede alvo para promover a estabilidade do treinamento, no Double DQN a rede principal é empregada exclusivamente para selecionar a ação mais vantajosa, enquanto a rede alvo é utilizada para avaliar o valor dessa ação, reduzindo a superestimação dos valores de Retorno (Van Hasselt *et al*., 2016). Sendo w_i e w^- os pesos da rede neural que faz a escolha da ação e da rede neural que avalia essa escolha, respectivamente, a função de perda utilizada no processo de treinamento é definida a cada iteração de acordo com o apresentado na Equação 9.

$$L_i(w_i) = \{[R(s,a,s') + \gamma Q(s', argmax_a, Q(s',a';w_i);w^-)] - Q(s,a;w_i)\}^2 \qquad (9)$$

Em que,

$argmax_a\, Q(s',a';w_i)$ = ação que maximiza o valor do Retorno no estado alcançado após executar a ação **a** no estado **s**, estimado com os pesos da rede neural da iteração atual.

$Q(s', argmax_a\, Q(s',a';w_i);w^-)$ = valor esperado do Retorno, estimado com os pesos da rede alvo, quando se é adotada a ação escolhida com $argmax_a\, Q(s',a';w_1)$ no estado **s'**.

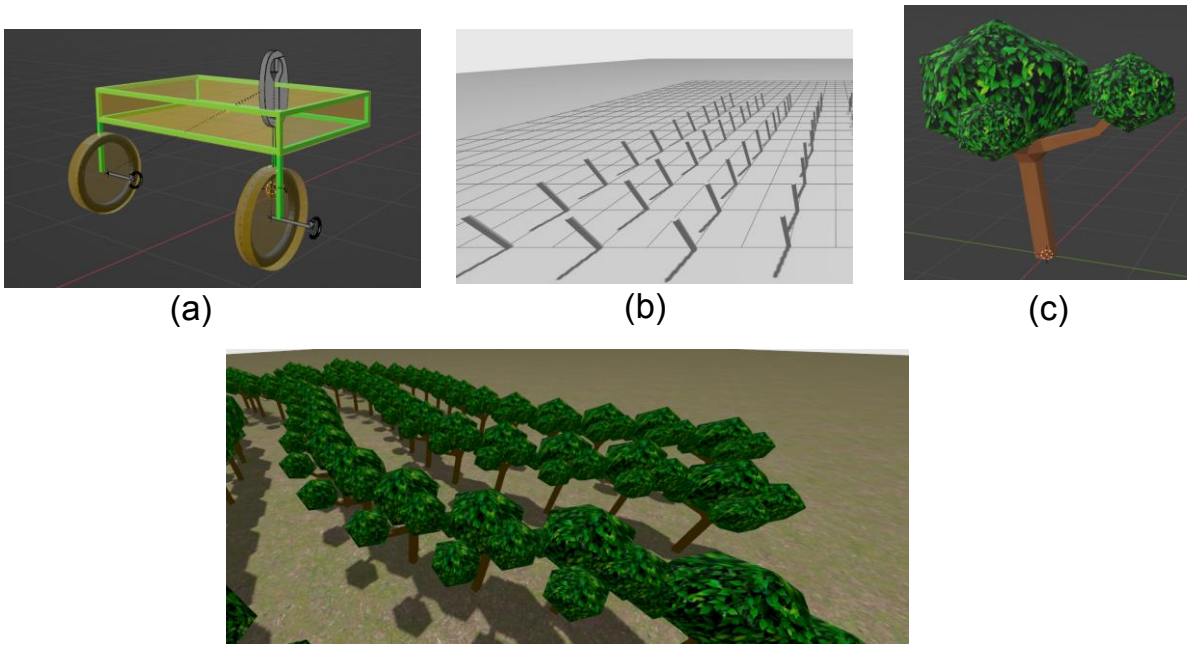
$Q(s,a;w_i)$ = previsão atual do valor do Retorno para a ação a no estado estimado com os pesos da rede neural da iteração atual.

3. Material e Métodos

O problema de *path tracking* foi modelado neste trabalho a partir de entrelinhas de plantio, que definiram trajetórias de referência a serem seguidas por uma plataforma robótica em simulações de áreas de plantio. Para isto, primeiramente a plataforma robótica e o ambiente de simulação foram modelados e desenvolvidos. Em seguida foram implementados dois modelos de controle, um controlador *Pure Pursuit* (PPC) e um sistema baseado em aprendizado por reforço. A aplicação do aprendizado por reforço à resolução do problema foi então avaliada em diferentes cenários de simulação, utilizando-se o PPC como referência comparativa.

3.1 Desenvolvimento do ambiente de simulação e da plataforma robótica

Na Figura 2 estão apresentados os componentes modelados para simulação da plataforma robótica e dos ambientes utilizados nos treinamentos e experimentos conduzidos.



(a) (b) (c)

(d)

Figura 2 – Representação das modelagens realizadas para condução das simulações: plataforma robótica (a); modelo físico de colisões (b); modelo visual das plantas (c); linhas de plantio geradas a partir da combinação dos modelos de colisão com o modelo visual (d).

A plataforma robótica utilizada no experimento foi modelada por meio do *software Blender*. O modelo foi desenvolvido para operar com um sistema de rotação diferencial, apresentando uma largura de 1,2 m, comprimento de 0,7 m, vão livre de 0,6 m, rodas com diâmetro de 0,4 m e velocidade angular máxima de 3 rad/s (Figura 2.a).

Para modelagem dos ambientes de simulação foi utilizado o *Robot Operating System 2* (ROS2), versão *Humble* (Macenski *et al.,* 2022). O ROS2 foi escolhido devido à sua robustez e flexibilidade na implementação de sistemas robóticos, permitindo a integração eficiente de sensores, atuadores e algoritmos de controle. Os ambientes foram implementados utilizando o *software Gazebo* (Koenig & Howard, 2004*)* versão *Ignition,* que possui integração com o ROS2 *Humble*. Nestes ambientes, os caules das plantas foram representados por cilindros com ruídos adicionados aos seus diâmetros e alturas (Figura 2.b). Além disso, foram adicionados à plataforma robótica os sensores GNSS, LiDAR e IMU, que estão disponíveis no sistema ROS2/Gazebo.

Um modelo visual fixo, também desenvolvido com o software Blender, foi utilizado com finalidade ilustrativa para representar a parte aérea das plantas (Figura 2.c). Com base nesses elementos, foram traçadas linhas de plantio (Figura 2.d), nas quais o espaçamento entre linhas e entre plantas pôde ser controlado de forma parametrizada.

O mundo virtual no Gazebo foi georreferenciado, tendo sua origem definida nas seguintes coordenadas: latitude -20,7547°, longitude -42,8739° e elevação de 648 metros, com base no datum WGS84 (*World Geodetic System* 1984). As trajetórias de referência para o robô, correspondentes às entrelinhas de cultivo, foram então definidas. Para as tarefas de navegação, essas trajetórias foram processadas no sistema de coordenadas projetado SIRGAS 2000 / UTM 23S.

As entrelinhas de deslocamento foram geradas individualmente, posicionadas no centro entre pares consecutivos de linhas de plantio. Cada entrelinha foi delimitada por pontos de referência localizados no início e no final de seu trajeto. Dessa forma, as áreas de deslocamento da plataforma foram compostas por um conjunto de entrelinhas centrais e trechos adicionais correspondentes às zonas de manobra, conforme ilustrado na Figura 3.

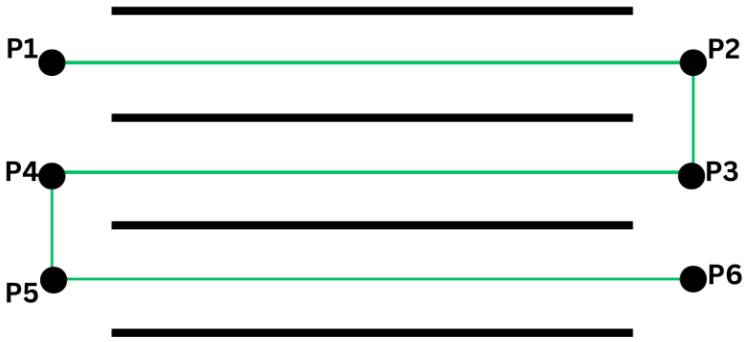


Figura 3 - Representação esquemática das linhas de plantio, entrelinhas de deslocamento e pontos de referência localizados no início e no fim de cada entrelinha.

Na área hipotética ilustrada na Figura 3, a plataforma robótica tem como objetivo deslocar-se do ponto P1 até o ponto P6. Para isso, deve percorrer sequencialmente cinco entrelinhas: de P1 a P2, P2 a P3, P3 a P4, P4 a P5 e, por fim, de P5 a P6. O ambiente de simulação foi projetado para atualizar dinamicamente a linha de deslocamento: sempre que a plataforma se aproximasse do ponto alvo atual dentro de uma distância pré-definida (10 centímetros), a próxima entrelinha era automaticamente ativada como nova referência de deslocamento.

Para garantir maior flexibilidade e representatividade no treinamento do algoritmo de aprendizado por reforço, desenvolveu-se um algoritmo para a geração dinâmica de áreas de simulação. Esse algoritmo permitiu a criação automatizada de ambientes de treinamento e validação, simulando diferentes configurações de plantio de maneira parametrizada. A principal motivação para essa abordagem foi a necessidade de testar a capacidade de aprendizado em diferentes cenários, evitando o sobreajuste a um único ambiente e aumentando a generalização do modelo treinado.

O algoritmo de geração de áreas utilizou ondas senoidais com perturbações estocásticas para simular linhas de plantio e entrelinhas de deslocamento com curvas suaves. As componentes x das coordenadas, que representam o espaçamento entre plantas, foram calculadas de forma iterativa para se obter a distância desejada ao longo da curva de plantio. As componentes das coordenadas y das plantas, por sua vez, foram obtidas por uma função que segue o formato apresentado na Equação 10 a seguir:

$$y(x) = Asin(Fx) + ce \quad (10)$$

Em que,

A = amplitude de ondulação da linha.

F = frequência de ondulação da linha.

c = índice da linha (que varia de 0 a n-1 linhas desejadas).

e = espaçamento entre linhas.

No escopo deste trabalho, as variáveis A e F foram definidas em função do comprimento das linhas de plantio e de fatores de forma (A_f e F_f), os quais são parâmetros de entrada do algoritmo. Além disso, foram aplicadas constantes obtidas a partir de análises visuais para que A_f e F_f próximos a um gerassem curvas visualmente suaves para diferentes comprimentos de linhas de plantio. Os cálculos de A e F em função de A_f e F_f estão apresentados nas Equações 11 e 12 a seguir:

$$A = 400\, L^{-1} A_f \quad (11)$$

$$F = 5\, L^{-1} F_f \quad (12)$$

Em que,

L = comprimento das linhas de plantio a serem traçadas

Para geração de uma área, o algoritmo utiliza os seguintes parâmetros: espaçamento médio entre linhas de plantio; espaçamento entre plantas em uma mesma linha; número de linhas de plantio; diâmetro do caules das plantas; altura das plantas; comprimento das linhas de plantio; fator de forma da amplitude das curvas; fator de frequência das curvas; distância de segurança entre as linhas de plantio e as zonas de manobra; ruído na posição transversal das plantas; ruído na posição inicial das linhas.

3.2 Algoritmo de aprendizado por reforço

Neste trabalho, as implementações do aprendizado por reforço foram feitas com a arquitetura Double DQN, que faz a integração do algoritmo Q-Learning com redes neurais profundas. O aprendizado por reforço é uma classe de algoritmos de aprendizado de máquina, utilizados para otimizar as escolhas de um agente em ambientes complexos. Conforme pode ser visto na Figura 4, a cada ação executada

o agente muda de estado, recebe *feedbacks* quantitativos (recompensas), e ajustar suas ações (avaliação) de forma a obter a máxima recompensa acumulada ao longo da interação com o ambiente.

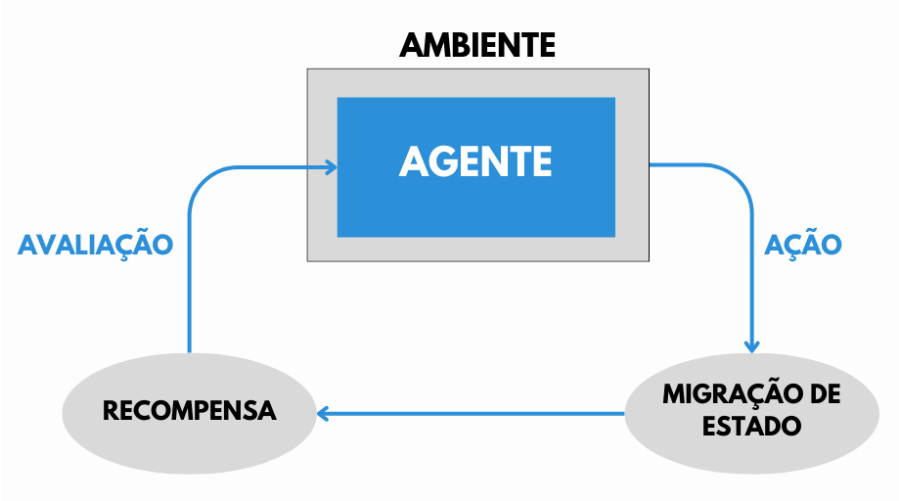


Figura 4 – Representação da interação do agente com o ambiente em um sistema de aprendizagem por reforço.

3.3 Modelagem do sistema de aprendizado por reforço

Neste trabalho, o processo de formulação do MDP correspondente ao problema de rastreamento de trajetória foi realizado por meio da implementação de nós no ROS2 que se comunicavam diretamente com os ambientes de simulação desenvolvidos no Gazebo. Estes nós foram responsáveis por coletar o estado do agente, enviar as ações escolhidas, calcular as recompensas obtidas a cada interação e executar o *loop* de treinamento do algoritmo de aprendizado por reforço.

3.3.1 Definição de estado do agente

Para definição do estado do agente foram utilizados os sensores GNSS, LiDAR e IMU, que estão disponíveis no sistema ROS2/Gazebo. O LiDAR foi configurado para coletar 1.000 pontos em uma varredura de 360 graus no plano horizontal, atingindo uma distância máxima de 5 m. Para fins de simplificação e redução da dimensionalidade dos dados, os 1.000 pontos foram agregados em 100, utilizando-se o valor mínimo de distância a cada grupo de 10 leituras consecutivas. O sistema GNSS foi configurado com base no modelo esférico WGS84 (*World Geodetic System*

1984). O sensor IMU foi configurado com origem no norte geográfico, medindo o ângulo no plano horizontal com relação a este referencial.

No sistema modelado, o estado do agente foi definido por um vetor formado por dados coletados por um ou mais dos três sensores configurados. A Figura 5 apresenta uma representação esquemática do processo de obtenção das componentes do vetor de estado provenientes dos sensores GNSS e IMU.

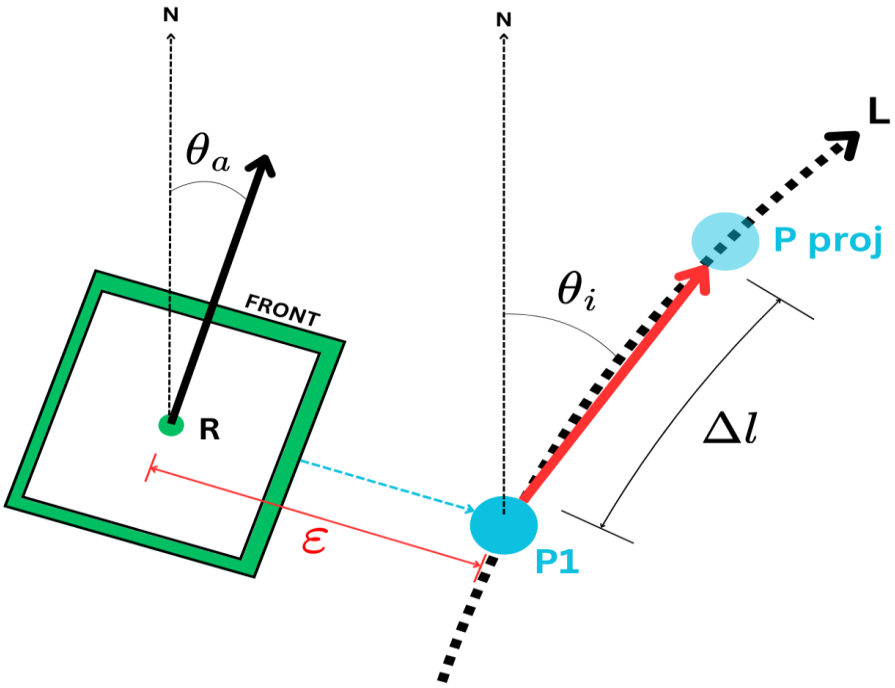


Figura 5 - Representação esquemática das componentes do vetor de estado do agente de aprendizado por reforço provenientes dos sensores GNSS e IMU.

Na Figura 5, a plataforma robótica (em verde) encontra-se na posição **R**. A linha **L,** também representada na figura, corresponde à entrelinha de deslocamento ideal previamente definida, cujo sentido está indicado na figura. O ponto **P1** é determinado como o ponto sobre a curva **L** mais próximo de **R** com base na distância euclidiana. A partir de **P1**, um ponto projetado **Pproj** é definido a uma distância l à frente, ao longo da curva **L**, seguindo o sentido da trajetória. O vetor que conecta **P1** a **Pproj**, (em vermelho na figura) define a direção de deslocamento de **P1** para **Pproj** em direção ao alvo. O ângulo θi é então calculado como o ângulo entre esse vetor e o norte geográfico, representando a orientação desejada. Por sua vez, o ângulo θa corresponde à orientação atual da plataforma em relação ao norte, medido pelo

sensor inercial (IMU). As componentes associadas à orientação são definidas como os valores de seno e cosseno dos ângulos θi (ideal) e θa (atual), permitindo ao agente distinguir direções mesmo sob periodicidade angular. Nos treinamentos e validações conduzidos neste trabalho, o valor de l utilizado foi de 0,5 m.

Outra componente do vetor de estado, que também pode ser observada na Figura 5, corresponde ao desvio lateral (ε), definido como a menor distância entre a posição atual da plataforma robótica, **R**, e a curva, **L**. Essa componente é construída a partir do valor absoluto do erro lateral, multiplicado por um fator de sinal (fs) que indica o lado da curva em que a plataforma se encontra. O fator fs assume o valor +1 quando a plataforma está à esquerda da entrelinha e −1 quando está à direita, permitindo que a rede neural diferencie desvios laterais de mesma magnitude, mas sentidos opostos. O cálculo de fs é realizado com base no produto vetorial entre o vetor tangente à curva ideal (que vai de **P1** a **Pproj**) e o vetor que vai da posição atual da plataforma, **R**, até o ponto mais próximo sobre a curva, **P1**, conforme descrito na Equação 13.

$$fs = sign(T \times D) = sign(T_x D_y - T_y D_x) \tag{13}$$

Onde,

$$T = Proj - P1$$
$$D = P1 - R$$

Todas as variáveis que compõem o vetor de estado foram normalizadas para o intervalo [−1,1], de modo a padronizar a escala das entradas da rede neural durante o processo de treinamento. Essa normalização foi realizada para evitar desbalanceamentos na magnitude das variáveis, promovendo maior estabilidade numérica e facilitando o aprendizado do modelo.

3.3.2 Definição do espaço de ações

O espaço de ações foi definido como um conjunto de pares de velocidades (v1,v2) aplicadas às rodas esquerda e direita da plataforma robótica, que opera por meio de um sistema de locomoção diferencial. O espaço de ações foi discretizado a

fim de viabilizar sua integração ao mecanismo de tomada de decisão do algoritmo DDQN. A velocidade angular máxima foi definida como 3 rad/s. Para realização de movimentos suaves, a diferença de rotação das duas rodas foi definida em um intervalo simétrico de -20% a 20% com passo de 1%, de forma que foram geradas 41 combinações de velocidades. Por fim, foram adicionadas três configurações específicas: uma de velocidade nula (0,0), e duas para rotação no próprio eixo (1.5, -1.5) e (-1.5,1.5), totalizando 44 combinações de velocidade dentro do espaço de ações.

3.3.3 Modelagem do sistema de recompensas

Um sistema customizado de recompensas foi desenvolvido para treinamento do modelo. Esse sistema foi modelado com o objetivo de reforçar ações que aproximavam o agente de um deslocamento ideal e penalizar comportamentos que levavam ao aumento dos desvios ou colisões. Os critérios de recompensa foram: distância deslocada até o próximo alvo, desvio lateral com relação à linha de deslocamento ideal e correção do erro angular. Além disso, foram aplicadas recompensas fixas associadas a colisões, desvios laterais muito grandes e ausência de movimentos (escolha persistente da velocidade zero para ambas as rodas).

Na Figura 6, tem-se a representação esquemática do deslocamento da plataforma robótica ao longo de uma entrelinha de deslocamento ideal, iniciando no ponto **X1** e avançando em direção ao ponto alvo **X2**. No instante inicial, a plataforma robótica encontra-se na posição **R1** e, após a execução de uma determinada ação, ela se desloca para uma nova posição **R2**.

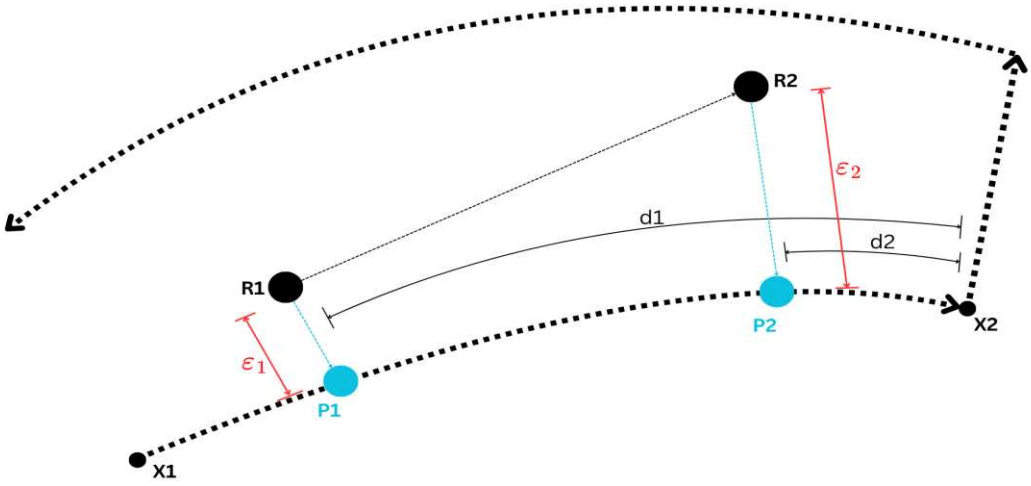


Figura 6 – Representação esquemática do deslocamento da plataforma robótica em relação à trajetória de referência. A linha pontilhada indica a trajetória ideal até o ponto alvo **X2**. Os pontos **R1** e **R2** representam posições reais consecutivas da plataforma, enquanto **P1** e **P2** correspondem às projeções ortogonais dessas posições sobre a trajetória de referência. As distâncias $\boldsymbol{\varepsilon_1}$e $\boldsymbol{\varepsilon_2}$ representam o erro lateral, e **d1** e **d2** indicam o comprimento do trajeto restante ao longo da trajetória ideal.

Na Figura 6, **P1** e **P2** representam os pontos, sobre a curva de deslocamento ideal, mais próximos de **R1** e **R2,** respectivamente. As distâncias **d1** e **d2** correspondem aos comprimentos do trajeto restante, ao longo da curva de deslocamento ideal, de **P1** e **P2** até o ponto alvo **X2**. A componente da recompensa associada ao progresso em direção ao alvo, denotada por **D**, é calculada pela diferença:

$$D = d_1 - d_2 \tag{14}$$

Esta componente representa o avanço efetivo da plataforma ao longo da trajetória planejada.

Quando a plataforma robótica se encontra na posição **R1** da Figura 6, o erro angular, denotado por $\theta_{\varepsilon 1}$ pode ser obtido a partir dos ângulos θ_i (ideal) e θ_a (atual), apresentados na Figura 5, a partir da subtração:

$$\theta_\varepsilon = \theta_i - \theta_a \tag{15}$$

O erro angular na posição **R2**, denotado por $\theta_{\varepsilon 2}$, também pode ser obtido com a Equação 15. A componente da recompensa associada à variação do erro angular, denotada por **A**, é obtida pela diferença entre os cossenos dos ângulos $\theta_{\varepsilon 2}$ e $\theta_{\varepsilon 1}$:

$$A = cos\,(\theta_{\varepsilon 2}) - cos\,(\theta_{\varepsilon 1}) \tag{16}$$

A componente da recompensa relacionada ao erro lateral, denotada por **ε** é definida como sendo o desvio lateral observado no estado para o qual a plataforma robótica migrou após a escolha de uma ação. Na Figura 6 este desvio é representado pela variável ε_2, na posição **R2**.

Para cálculo das recompensas, as componentes **D**, **A** e **ε** foram normalizadas no intervalo [-1,1] a partir da divisão dessas variáveis pelos valores máximos, em módulo, que elas poderiam assumir. Após essa normalização, cada componente foi multiplicado por um coeficiente específico, utilizado para ajustar sua contribuição relativa na recompensa total. Além disso, foi aplicado um incremento de 25% no peso das componentes quando os valores normalizados se encontravam no intervalo [-1,0), de forma que, na ocorrência de movimentos oscilatórios persistentes a recompensa não seria nula, mas sim negativa.

Um valor fixo e negativo foi acrescentado à recompensa final sempre que o episódio era encerrado devido a uma das condições de parada descritas nos itens *ii, iii ou iv* da subseção 3.3.4. Essa penalização adicional teve como objetivo desencorajar comportamentos indesejados, como colisões, desvios excessivos ou inatividade prolongada.

Desta forma, a cada ação executada, a recompensa total obtida (Rw) foi calculada conforme apresentado na Equação (17).

$$Rw = M + \alpha_\varepsilon\, \varepsilon + c\, k_p \tag{17}$$

Onde,

Rw = recompensa total obtida.

c = constante associada à ocorrência de fenômenos terminais indesejados.

- c = 1 na ocorrência das condições de parada descritas nos itens *ii, iii* ou *iv* da subseção 3.3.4.

- c = 0 quando nenhuma dessas condições é satisfeita.

M = variável condicional, que pode ser igual à recompensa angular ou à recompensa relacionada com o progresso em direção ao alvo.

ε = componente de recompensa relacionada ao desvio lateral.

α_ε = coeficiente de contribuição associado à componente de desvio lateral.

k_p = punição aplicada em caso de colisão, desvio maior que 1 m ou inatividade prolongada.

A variável **M** foi determinada com base na orientação da plataforma robótica em relação à direção do vetor de deslocamento ideal (Figura 7) no instante anterior à execução da ação (ponto **R1** representado na Figura 6).

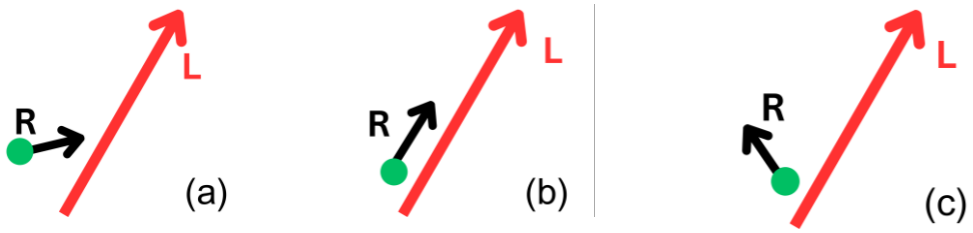


Figura 7 - Representação do vetor de orientação da plataforma robótica (R) e do vetor de deslocamento ideal sobre a entrelinha de deslocamento (L).

Nos casos em que a plataforma robótica se encontrava orientada em direção ao vetor de deslocamento ideal **L** (Figura 7.a) ou paralela a este vetor (Figura 7.b), priorizou-se o avanço em direção ao alvo, de forma que **M** assumia o valor da componente associada ao progresso de deslocamento (**D**) ponderada por seu respectivo coeficiente de contribuição. Para considerar o deslocamento como paralelo ao vetor ideal, adotou-se um pequeno limiar, permitindo que valores de erro angular próximos a zero ainda fossem classificados como paralelos.

Por outro lado, nos casos em que o vetor de deslocamento da plataforma estava em direção oposta ao vetor de deslocamento ideal (Figura 7.c), a prioridade foi dada à correção do erro angular, de forma que **M** assumia o valor da componente angular (**A**), também ponderada por seu coeficiente de contribuição.

A seguir estão apresentados os valores utilizados para as constantes utilizadas no cálculo da recompensa, obtidas de forma experimental:

- Punição aplicada em caso de colisão, desvio maior que 1 m ou inatividade prolongada (k_p): -1
- Coeficiente de contribuição associado ao componente de progresso em direção ao alvo (α_D): +0,15
- Coeficiente de contribuição associado à correção do erro angular (α_A): +0,15
- Coeficiente de contribuição associado ao desvio lateral (**ε**): -0,5
- Limiar de deslocamento paralelo: 0.1 rad

3.3.4 Parâmetros de treinamento

Os treinamentos foram conduzidos de forma que cada episódio era encerrado automaticamente ao se atingir uma das seguintes condições de parada:

i) Conclusão bem-sucedida do trajeto, quando a plataforma robótica percorria integralmente a área de treinamento até o ponto final designado;

ii) Colisão com uma linha de plantio, detectada com base na distância registrada pelo sensor LiDAR em relação ao obstáculo mais próximo (considerou-se colisão sempre que um objeto se encontrava a uma distância menor que 75 cm do centro da plataforma robótica);

iii) Desvio lateral superior a 1 m;

iv) Inatividade prolongada, caracterizada pela escolha recorrente da ação correspondente à velocidade zero em ambas as rodas, implicando que o agente optou por permanecer parado por um período excessivo.

O ambiente de simulação foi configurado para operar a uma frequência de 10 Hz, de modo que, a cada segundo, o ciclo completo de leitura do estado atual, seleção da ação, armazenamento da transição de estado e retropropagação da rede neural foi executado dez vezes. Para armazenar as experiências do agente, utilizou-se uma memória de replay (*replay buffer*) com capacidade para 100 mil pontos, contendo informações sobre o estado inicial, a ação escolhida, o novo estado e a recompensa obtida.

O estado do agente foi definido como um vetor de tamanho 17, sendo 2 componentes correspondentes ao seno e ao cosseno ângulo registrado pelo IMU, e 15 componentes correspondentes ao seno, o cosseno e o desvio lateral obtidos a

partir das 5 últimas leituras do sistema GNSS. Estas variáveis foram obtidas conforme apresentado na subseção 3.3.1.

O valor do fator de desconto utilizado foi de 0,99. A rede neural utilizada como função de aproximação no algoritmo DDQN foi estruturada como uma arquitetura do tipo *feedforward*. A camada de entrada contou com 17 neurônios, quantidade de correspondentes às variáveis do vetor de estado do agente. Em seguida, foram utilizadas três camadas ocultas com 64, 32 e 16 neurônios, respectivamente. Por fim, a camada de saída foi composta por 44 neurônios, cada um representando uma das ações possíveis no espaço de ações discretizado. Para a ativação das camadas ocultas, foi empregada a função ReLU (*Rectified Linear Unit*). O processo de retropropagação foi conduzido utilizando um *batch size* de 128 amostras, selecionadas aleatoriamente do *replay buffer*. As amostragens foram realizadas sem reposição, garantindo que cada transição fosse utilizada apenas uma vez por iteração. Foi utilizado um processo de atualização suave dos pesos da rede neural alvo (*soft update*) (Equação 18) com um coeficiente de suavização (τ) de 0,001, de forma que, a cada iteração, os pesos da rede-alvo foram atualizados em 0,1% na direção dos valores correspondentes na rede de avaliação.

$$\theta_{target} \leftarrow \tau\theta_{eval} + (1-\tau)\theta_{target} \tag{18}$$

Em que,

θ_{target} = vetor de pesos da rede alvo.

θ_{target} = vetor de pesos da rede de avaliação.

τ = coeficiente de suavização.

Os treinamentos foram conduzidos com a estratégia de exploração *ε-greedy*. Com esta estratégia, a cada iteração a ação escolhida pelo agente tinha uma probabilidade ε de ser escolhida de forma aleatória e uma probabilidade (1 - ε) de ser escolhida pelo algoritmo DDQN. No início do treinamento, teve-se a utilização de um ε igual a 1, de forma que as ações foram escolhidas de forma completamente aleatórias e o ambiente foi explorado. À medida que o algoritmo foi treinado com a experiências armazenadas no *replay buffer*, teve-se uma diminuição do valor de ε. Neste trabalho, utilizou-se uma diminuição fixa de 10^{-5}. O valor mínimo de ε foi definido

como 0,05 de forma que, ao longo de todo o treinamento, fossem utilizados pelos menos 5% de exploração. Com esta configuração, o valor de ε diminuiu 1 para 0,05 em 95 mil iterações.

O agente foi submetido a um processo de treinamento em duas etapas. Na primeira, os sensores foram configurados com erros de medição compatíveis com valores típicos de operação de alta precisão, utilizando distribuições normais com média zero e desvio padrão correspondente ao erro simulado - IMU: 0,5°; GNSS: 0,05 m; patinagem: 5%. Utilizou-se uma taxa de aprendizado de 10^{-4}, mantendo-se o treinamento até a estabilização do score. Na segunda etapa, adotou-se uma taxa de aprendizado reduzida (10^{-5}) e incorporaram-se falhas aleatórias no sensor GNSS, variando entre 1 e 10 m, com probabilidade de 0,1% a cada ação executada. O modelo resultante desse processo foi utilizado em todas as avaliações apresentadas neste trabalho.

Os treinamentos e experimentos foram conduzidos em um ambiente computacional equipado com uma unidade de processamento central (CPU) Intel Core i7-13620H, composta por 10 núcleos e 16 threads. A unidade de processamento gráfico (GPU) utilizada foi uma NVIDIA GeForce RTX 4050 com 6 GB de memória GDDR6. O sistema dispunha de 16 GB de memória RAM DDR5. As implementações dos algoritmos de aprendizado por reforço foram desenvolvidas em Python 3.10, utilizando a biblioteca PyTorch 2.1, em ambiente Linux (Ubuntu 22.04 LTS).

3.4 Algoritmo Pure Pursuit

Para servir como *baseline* de comparação, foi implementado o controlador de seguimento de trajetória *Pure Pursuit* (PPC), aplicado à mesma plataforma robótica e sob as mesmas condições dos experimentos com aprendizado por reforço. O PPC é um algoritmo de controle cinemático baseado em geometria, amplamente adotado em robótica móvel devido à sua simplicidade, intuição e eficácia em uma variedade de cenários de navegação. A implementação se deu com base no apresentado por Coulter (1992).

O controlador foi implementado no ambiente ROS 2, utilizando dados de orientação fornecidos pelo sensor IMU e posição geográfica obtida via sensor GNSS. A distância entre as rodas motrizes (*wheelbase*) foi definida como 1,2 m, conforme as dimensões da plataforma robótica. A velocidade base adotada foi de 0,6 m/s,

equivalente à velocidade angular de 3 rad/s utilizada também no sistema de aprendizado por reforço. O parâmetro de antecipação (*lookahead distance*) foi ajustado por meio de testes, levando em consideração os níveis de ruído introduzidos no sensor GNSS e buscando o equilíbrio entre a estabilidade do controle e a precisão no seguimento da trajetória. Nos experimentos conduzidos o valor utilizado foi de 0,5 m. As velocidades lineares e angulares calculadas pelo algoritmo foram publicadas diretamente no nó de controle das rodas da plataforma robótica.

O PPC foi utilizado como referência para a avaliação do desempenho do agente treinado por aprendizado por reforço, permitindo uma análise comparativa qualitativa e quantitativa da capacidade de rastreamento de trajetória em diferentes cenários simulados.

3.5 Experimentos e parâmetros

Para avaliar o sistema desenvolvido para geração de ambientes de simulação, o algoritmo de geração de áreas proposto foi utilizado na criação de diferentes áreas virtuais, com variação nos parâmetros disponíveis. Três ambientes distintos foram gerados para os experimentos com o algoritmo DDQN. A seguir, detalha-se a configuração das áreas utilizadas para treinamento e validação do modelo:

i) - Área de treinamento (A0)

- Fatores de Forma: $A_f = 1.5$ e $F_f = 1.5$
- Configuração das linhas de plantio: espaçamento entre linhas de 2,0 m; espaçamento entre plantas de 1,0 m; 10 linhas de deslocamento com 15 m de comprimento.
- Representação dos caules das plantas: caules com diâmetro de 0,1 m e altura de 1,0 m.

ii) - Área de avaliação 1 (A1)

- Fatores de Forma: $A_f = 2.4$ e $F_f = 1.2$
- Configuração das linhas de plantio: espaçamento entre linhas de 2,5 m; espaçamento entre plantas de 0,5 m; 6 linhas de deslocamento com 20 m de comprimento.

- Representação dos caules das plantas: caules com diâmetro de 0,05 m e altura de 1,0 m.

iii) - Área de avaliação 2 (A2)

- Fatores de Forma: $A_f = 0$ e $F_f = 0$
- Configuração das linhas de plantio: espaçamento entre linhas de 2,0 m; espaçamento entre plantas de 1,5 m; 6 linhas de deslocamento com 20 m de comprimento.
- Representação dos caules das plantas: caules com diâmetro de 0,2 m e altura de 1,0 m.

Às três áreas geradas, procedeu-se à adição dos seguintes ruídos aleatórios, utilizando-se distribuição uniforme:

- Deslocamento inicial das linhas de plantio: ±0,5 m;
- Variação no espaçamento entre linhas: ±0,1 m;
- Variação no espaçamento entre plantas: ±0,1 m;
- Ruído no diâmetro e na altura dos caules: ±1%.

Para condução dos experimentos, utilizou-se uma simulação simplificada de patinagem por meio de um desconto aleatório aplicado nas rotações enviadas para os pneus da plataforma robótica. Para isso, utilizou-se uma distribuição normal truncada de zero a infinito (*Truncated Normal*) com média de 5% e desvio padrão de 1%. Para simulação dos erros nos sensores foram utilizadas distribuições normais com média zero e desvio padrão correspondente ao erro simulado. No sensor IMU, o valor utilizado foi de 0,5 graus; no GNSS o valor foi de 0,05 m.

Por fim, para simulação de falhas no GNSS, utilizou-se a alteração, a uma dada probabilidade e por um período de tempo definido, do valor do desvio padrão incorporado ao erro. Para cálculo da probabilidade de falha a cada iteração, utilizou-se uma equação baseada na distribuição de probabilidades discreta binomial, conforme apresentado na Equação 19.

$$p_i = 1 - exp(log(1 - p_{falha})\,(f.t)^{-1}) \tag{19}$$

Em que,

p_i = probabilidade de ocorrência de falha em uma iteração.

p_{falha} = probabilidade desejada de falha no GNSS.

f = frequência de atuação do controlador.

t= Tempo máximo de falha desejado.

Na implementação realizada, as falhas foram aplicadas por um período de tempo aleatório de um segundo até o tempo máximo de falha desejado, escolhido por meio de uma distribuição uniforme.

A partir destes parâmetros, dois cenários de experimentos foram abordados neste trabalho:

i) Ausência de falhas (G1) – erro do sensor GNSS com desvio padrão de 0,05 m em 100% do tempo

ii) Falhas abruptas (G2) – erro do sensor GNSS com desvio padrão de 0,05 m com uma probabilidade de 20% de ocorrência de um erro com desvio padrão de 10 m a cada 5 segundos por um período de tempo aleatório de 1 a 5 segundos.

A partir dos parâmetros apresentados, 12 cenários experimentais foram formados combinando-se os dois modelos (DDQN e PPC), as três áreas (A0, A1 e A2) e dois cenários de erro do GNSS (G1 e G2).

Em cada cenário experimental, para avaliação da capacidade dos modelos de evitar colisões, foram executadas 30 repetições com o modelo físico de colisões das plantas (Figura 2.d). Adicionalmente, foram conduzidas 30 repetições em ambiente sem o modelo físico de colisões, com o objetivo de isolar e analisar o desempenho do deslocamento do robô, do ponto inicial ao final da trajetória, sem que houvesse o encerramento da simulação pela ocorrência de impactos.

Para a análise estatística dos resultados, foi adotada uma abordagem não-paramétrica para garantir a robustez das conclusões, dado que a premissa de normalidade (avaliada pelo teste de Shapiro-Wilk) não foi satisfeita para todas as métricas e grupos experimentais. As comparações de desempenho entre os controladores para cada condição específica foram realizadas utilizando o teste bilateral de Mann-Whitney U, com um nível de significância de $\alpha = 0.05$.

4. Resultados e Discussão

4.1 Áreas de treinamento e validação

Na Figura 8 podem ser observadas as três áreas georreferenciadas (8.a, 8.c e 8.e) e modeladas no ambiente *Gazebo (8.b, 8.d e 8.f)* geradas para treinamento e validação dos modelos.

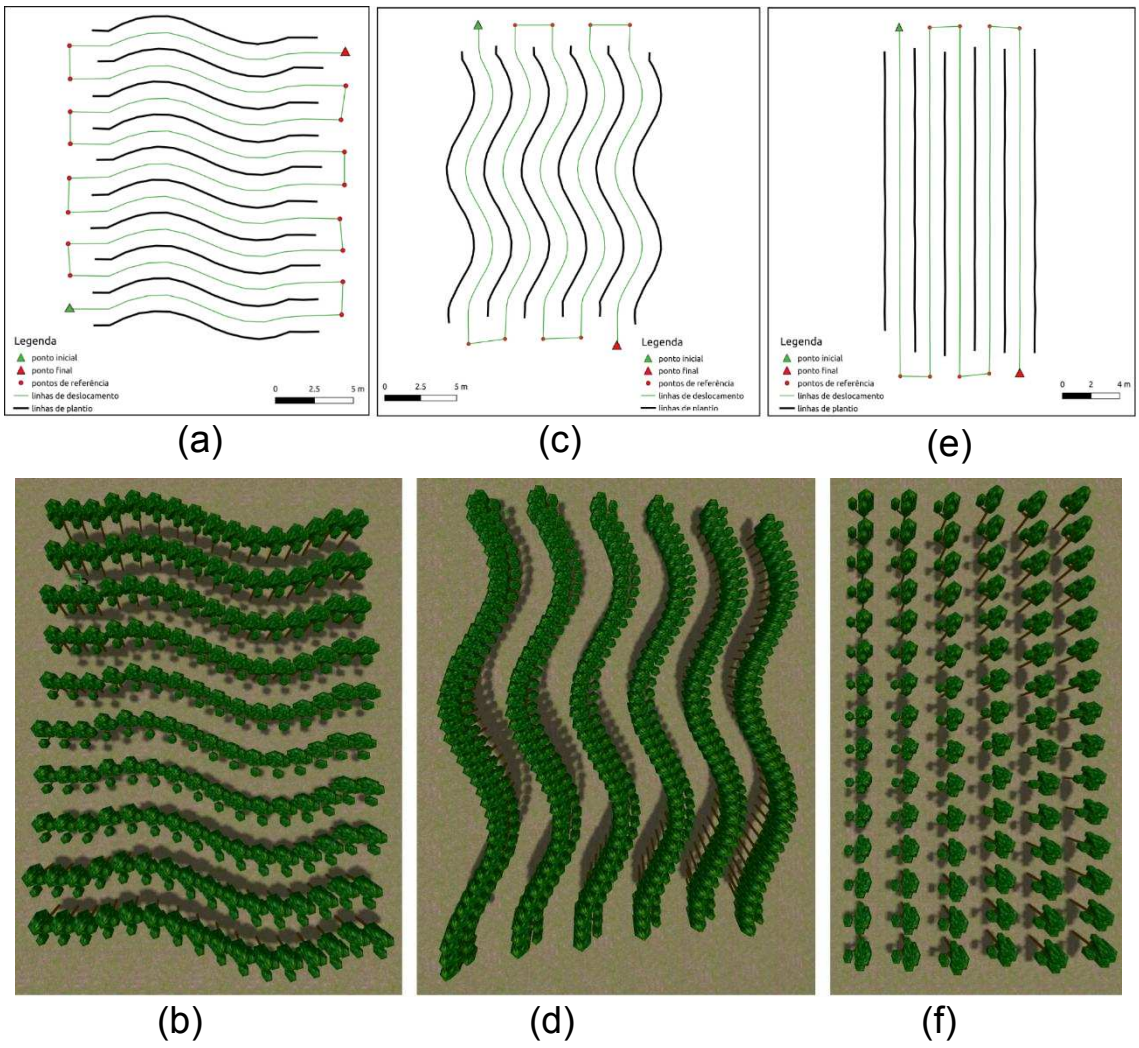


Figura 8 – Representação, na forma georreferenciada e na modelagem feita no ambiente Gazebo, das áreas utilizadas para treinamento dos modelos e condução dos experimentos. Em 8.a e 8.b tem-se a área utilizada para treinamento do modelo de aprendizado por reforço (A0). Em 8.c e 8.d, tem-se a área de validação com curvas (A1) e em 8.e e 8.f a área de validação com linhas retas (A2).

Conforme ilustrado na Figura 8, as áreas geradas demonstraram conformidade com os parâmetros configurados no algoritmo de geração dinâmica, tanto em termos de dimensões geométricas quanto na inserção dos ruídos controlados. Os resultados evidenciam que o algoritmo proposto se apresentou como uma solução eficiente, ágil

e flexível para a criação de ambientes virtuais destinados ao treinamento e à validação de modelos de controle autônomo em sistemas robóticos agrícolas.

4.2 Análise de Desempenho em Navegação de Alta Precisão (Cenário G1: sem Falhas)

No cenário G1 (sem falhas), os controladores PPC e DDQN completaram 100% das 30 repetições em A0, A1 e A2 sem colisões, indicando robustez de ambos os controladores frente ao ruído dos sensores e à simulação estocástica patinagem.

A Tabela 1 apresenta as médias e os desvios padrão de três métricas de erro de trajetória analisadas: erro absoluto médio (MAE), erro absoluto máximo, e raiz quadrada do erro quadrático médio (RMSE), calculados para os seis cenários experimentais sem falhas no sistema GNSS (cenário G1). As comparações estatísticas entre os controladores DDQN e PPC são indicadas por letras de significância. Dentro de cada combinação de área e métrica avaliada, valores que não compartilham a mesma letra, na mesma linha, diferem estatisticamente quanto ao desempenho (teste de Mann–Whitney U, $p < 0,05$).

Tabela 1 - médias e os desvios padrão de três métricas de erro da trajetória analisadas nas três áreas avaliadas (A0, A1 e A2).

	MAE (cm)		Erro absoluto máximo (cm)		RMSE (cm)	
	DDQN	PPC	DDQN	PPC	DDQN	PPC
A0	3,9 ± 0,1 ***a***	4,2 ± 0,2 ***b***	14,0 ± 0,8 ***c***	32,0 ± 2,8 ***d***	4,8 ± 0,1 ***e***	7,6 ± 0,4 ***f***
A1	6,0 ± 0,2 ***b***	3,3 ± 0,2 ***a***	26,0 ± 1,3 ***c***	29,4 ± 3,6 ***d***	7,8 ± 0,2 ***f***	6,4 ± 0,4 ***e***
A2	4,4 ± 0,2 ***b***	3,5 ± 0,2 ***a***	16,5 ± 2,1 ***c***	30,8 ± 3,7 ***d***	5,2 ± 0,3 ***e***	6,6 ± 0,5 ***f***

Legenda: MAE = Erro absoluto médio; RMSE = Raiz quadrada do erro quadrático médio; DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.

Dentro de cada combinação de área e métrica avaliada, médias seguidas da mesma letra não diferem pelo teste de Mann–Whitney U ($p < 0,05$).

Conforme apresentado na Tabela 1, foram observadas diferenças estatisticamente significativas nos valores de MAE entre os modelos DDQN e PPC nas três áreas avaliadas. O DDQN obteve menor MAE na área de treinamento (A0),

indicando um desempenho mais preciso nesse ambiente. Por outro lado, nas áreas de validação (A1 e A2), o PPC apresentou valores de MAE inferiores. Esse padrão pode sugerir um leve indício de sobreajuste do DDQN ao ambiente de treinamento, especialmente em termos de minimização de desvios em trajetórias previamente exploradas. No entanto, tal efeito não comprometeu sua capacidade de generalização funcional, uma vez que, mesmo apresentando MAEs superiores nas áreas de validação, o DDQN foi capaz de realizar todos os deslocamentos sem colisões — um indicativo robusto de desempenho adequado sob condições não vistas.

Em relação ao erro absoluto máximo, observa-se na Tabela 1 que o modelo DDQN apresentou desempenho estatisticamente superior ao PPC nas três áreas avaliadas. Para essa métrica, a menor média do DDQN foi registrada na área de treinamento. Nesta área, o DDQN superou o PPC com uma diferença de 56%, considerando a média do erro máximo. Nas áreas A1 e A2, a redução foi de 11% e 46%, respectivamente. Esses resultados indicam que o DDQN obteve ganhos expressivos nas áreas A0 e A2 em termos de desvios extremos.

Na Figura 9 são apresentadas as posições registradas pelo sistema GNSS ao longo das 30 repetições para os modelos DDQN (Figura 9.a) e PPC (Figura 9.b) na área de teste (A0), no qual se verificou a maior diferença entre os modelos quanto ao erro absoluto máximo. Nessa Figura, estão representadas as posições reais da plataforma robótica, obtidas antes da adição do ruído gaussiano com desvio-padrão de 5 cm.

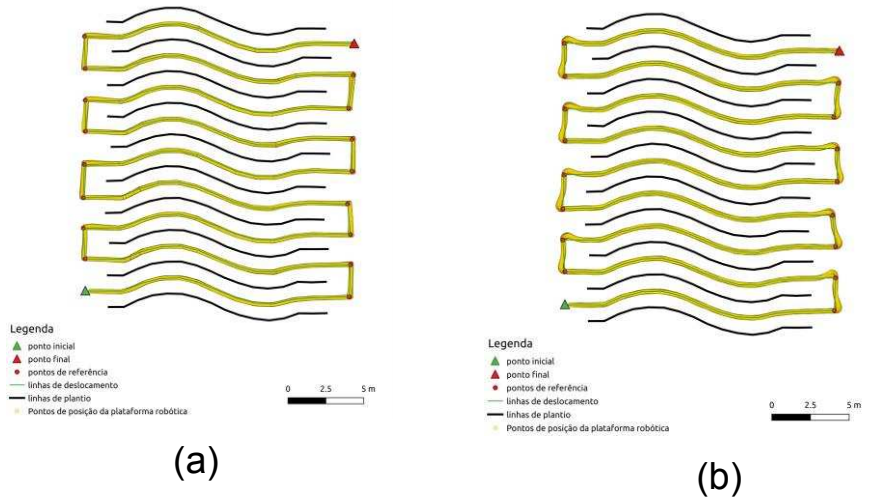


Figura 9 – Distribuição espacial dos pontos de posição registrados pela plataforma robótica ao longo de 30 repetições para o modelo DDQN (a) e para o PPC (b).

As distribuições de pontos apresentadas na Figura 9 ilustram visualmente a uniformidade dos deslocamentos descrita na Tabela 1, evidenciando baixa variação ao longo das 30 repetições para ambos os modelos. Observa-se ainda que, no caso do PPC (Figura 9.b), os maiores desvios concentraram-se nas regiões de mudança de linha de deslocamento. Esse comportamento é condizente com o encontrado na literatura, que identifica a correção de ângulos bruscos como uma das principais limitações do modelo (Yao *et al*., 2020). Tal limitação justifica o fato de o DDQN ter superado o PPC em todos os cenários no quesito desvio absoluto máximo, indicando que o DDQN desenvolveu uma política mais eficaz para a correção dessas transições abruptas.

No que se refere ao RMSE do erro absoluto, os resultados da Tabela 1 indicam que o DDQN apresentou desempenho superior ao PPC nas áreas A0 e A2, enquanto o PPC foi superior em A1. Todas as diferenças foram estatisticamente significativas.

A sensibilidade do RMSE a valores extremos permite interpretar essas diferenças de forma mais detalhada. Tanto em A1 quanto em A2, o DDQN apresentou menor média do desvio absoluto máximo e maior média do MAE em comparação ao PPC. Em A1, a média do RMSE do PPC foi menor, sugerindo que os erros mais frequentes tiveram maior peso no resultado global do que os erros isolados de maior magnitude. Em contraste, na área A2, o DDQN obteve RMSE menor, o que sugere que os erros extremos do PPC exerceram maior influência sobre essa métrica.

Os resultados observados indicam que o agente DDQN foi capaz de aprender uma política estável e robusta, demonstrando não apenas desempenho satisfatório nas áreas utilizadas durante o treinamento, mas também uma significativa capacidade de generalização para regiões não exploradas anteriormente. O principal diferencial do DDQN em relação ao PPC corresponde a redução expressiva da ocorrência de desvios extremos, especialmente em zonas que demandam manobras complexas e precisas.

4.3 Análise de Robustez sob Falhas de GNSS (Cenário G2: Com Falha)

Na Tabela 2 estão apresentadas as médias, os desvios padrão e os coeficientes de variação do percentual do trajeto total concluído em cada uma das áreas, nas 30 repetições, com os dois modelos no cenário de falhas do GNSS (G2).

As comparações estatísticas (Teste de Mann-Whitney U, $p < 0,05$) entre os controladores para cada área são indicadas por letras de significância.

Tabela 2 - Médias, desvios padrão e coeficientes de variação do percentual dos trajetos concluídos sem colisões por cada modelo em cada uma das áreas analisadas (A0, A1 e A2) nas 30 repetições, no cenário de falhas do sensor GNSS.

	Percentual concluído (%)		Coeficiente de variação (%)	
	DDQN	PPC	DDQN	PPC
A0	63,0 ± 18,9 ***a***	38,5 ± 29,3 ***b***	30,0	76,1
A1	81,2 ± 25,9 ***a***	70,1 ± 32,4 ***a***	29,5	46,2
A2	92,2 ± 10,6 ***a***	64,4 ± 34,8 ***b***	11,5	54,0

Legenda: DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.
Dentro de cada combinação de área e métrica avaliada, médias seguidas da mesma letra não diferem pelo teste de Mann–Whitney U ($p < 0,05$).

A análise dos resultados apresentados na Tabela 2 mostra que o agente DDQN obteve maiores médias de percentuais de trajetos concluídos sem colisões sob falhas bruscas do sinal do sistema GNSS em todas as áreas avaliadas. As diferenças entre os modelos foram estatisticamente significativas em A0 e A2, evidenciando desempenho consistentemente melhor do DDQN nestes cenários.

Além do desempenho médio superior, o DDQN apresentou menor variabilidade operacional, conforme indicado pelos menores coeficientes de variação em todas as áreas. Em contrapartida, o PPC exibiu variabilidade acentuada. Na área A0, por exemplo, o coeficiente de variação do PPC atingiu 76,1%, sugerindo ampla dispersão dos pontos de colisão ao longo do trajeto. O DDQN, nessa mesma área, apresentou CV de 30%, evidenciando maior estabilidade no controle de navegação.

Na Figura 10 está apresentada a dispersão dos percentuais concluídos nas 30 repetições para ambos os modelos nas três áreas analisadas.

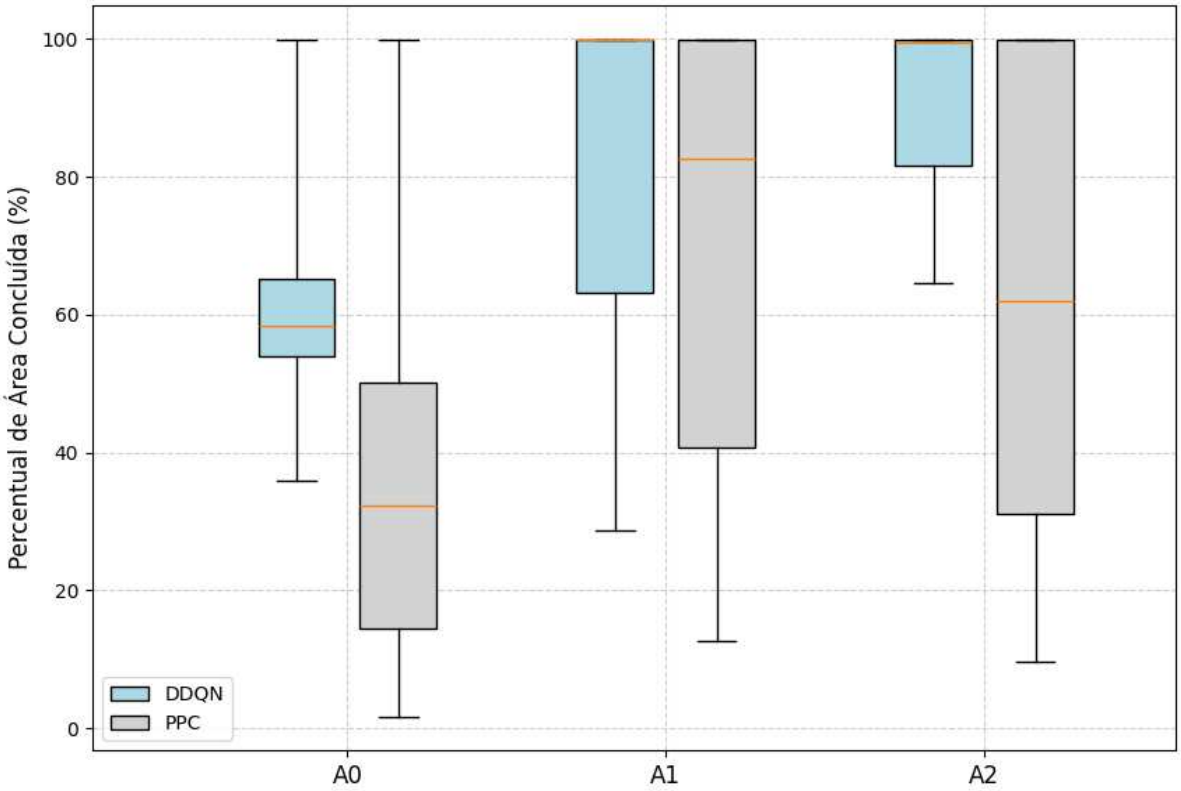


Figura 10 - percentuais das áreas deslocados sem colisão nas 30 repetições para ambos os modelos nas três áreas analisadas.

Conforme apresentado na Figura 10, em todas as áreas avaliadas, a mediana do DDQN manteve-se superior à do PPC. Nas áreas A1 e A2, a mediana coincidiu com o valor máximo, indicando que, em mais de 50 % das repetições, o controlador DDQN completou o percurso integral da área sem ocorrência de colisões. Além disso, observa-se que o intervalo interquartílico do DDQN foi consistentemente menor, evidenciando maior homogeneidade e previsibilidade no desempenho entre as repetições.

Em A0, o PPC exibe ampla dispersão e valores mínimos próximos a zero, indicando elevada frequência de falhas precoces, enquanto o DDQN mantém um patamar mínimo substancialmente mais alto e variabilidade reduzida. Nesta área ocorreram os menores percentuais de conclusão para ambos os modelos, o que pode ser atribuído ao seu percurso mais extenso, composto por linhas de menor comprimento e maior proporção de zonas de manobra em relação à distância total percorrida. Essa configuração tende a aumentar a frequência de mudanças de direção e, consequentemente, as oportunidades de falha.

Em A1, embora a mediana do DDQN seja superior, as distribuições apresentam sobreposição considerável, o que se alinha à ausência de diferença estatística significativa. Em A2, a vantagem do DDQN se reflete tanto na mediana elevada quanto na menor amplitude interquartílica, sugerindo maior previsibilidade operacional.

Ao comparar A1 e A2 — áreas de mesmo comprimento total, mas características distintas de geometria e espaçamento — observa-se que o DDQN obtém seu melhor desempenho em A2, enquanto o PPC apresenta seu melhor resultado em A1. A análise sugere que o DDQN é mais sensível à complexidade geométrica, sofrendo maior degradação em áreas com curvas acentuadas, enquanto o PPC é mais vulnerável à densidade de obstáculos, especialmente quando o espaçamento entre plantas é reduzido. Em teoria, a configuração retilínea e previsível de A2 favorece ambos os controladores; entretanto, o menor espaçamento entre linhas parece ter limitado o ganho de desempenho do PPC, ao passo que o DDQN conseguiu explorar plenamente a simplicidade geométrica para atingir valores próximos ao ideal.

A Tabela 3 apresenta as médias e desvios padrão das três métricas de erro de trajetória analisadas nos cenários com falha do sensor GNSS, considerando a área livre (sem modelos físicos de colisão com as plantas). As letras de significância indicam diferenças estatísticas segundo o teste de Mann-Whitney U ($p < 0,05$).

Tabela 3 - médias e os desvios padrão de três métricas de erro da trajetória analisadas no cenário de falhas de GNSS nas três áreas analisadas (A0, A1 e A2).

	MAE (cm)		Erro absoluto máximo (cm)		RMSE (cm)	
	DDQN	PPC	DDQN	PPC	DDQN	PPC
A0	7,2 ± 0,6 ***a***	9,4 ± 4,0 ***b***	30,9 ± 10,6 ***c***	127,4 ± 76,2 ***d***	9,4 ± 1,2 ***e***	22,0 ± 12,2 ***f***
A1	7,4 ± 0,7 ***b***	5,2 ± 1,4 ***a***	32,7 ± 8,9 ***c***	49,8 ± 22,9 ***d***	9,9 ± 1,5 ***e***	10,3 ± 3,7 ***e***
A2	6,1 ± 1,5 ***a***	6,1 ± 1,8 ***a***	43,5 ± 14,8 ***c***	62,0 ± 23,5 ***d***	9,3 ± 3,0 ***e***	12,3 ± 4,4 ***f***

Legenda: MAE = Erro absoluto médio; RMSE = Raiz quadrada do erro quadrático médio; DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.

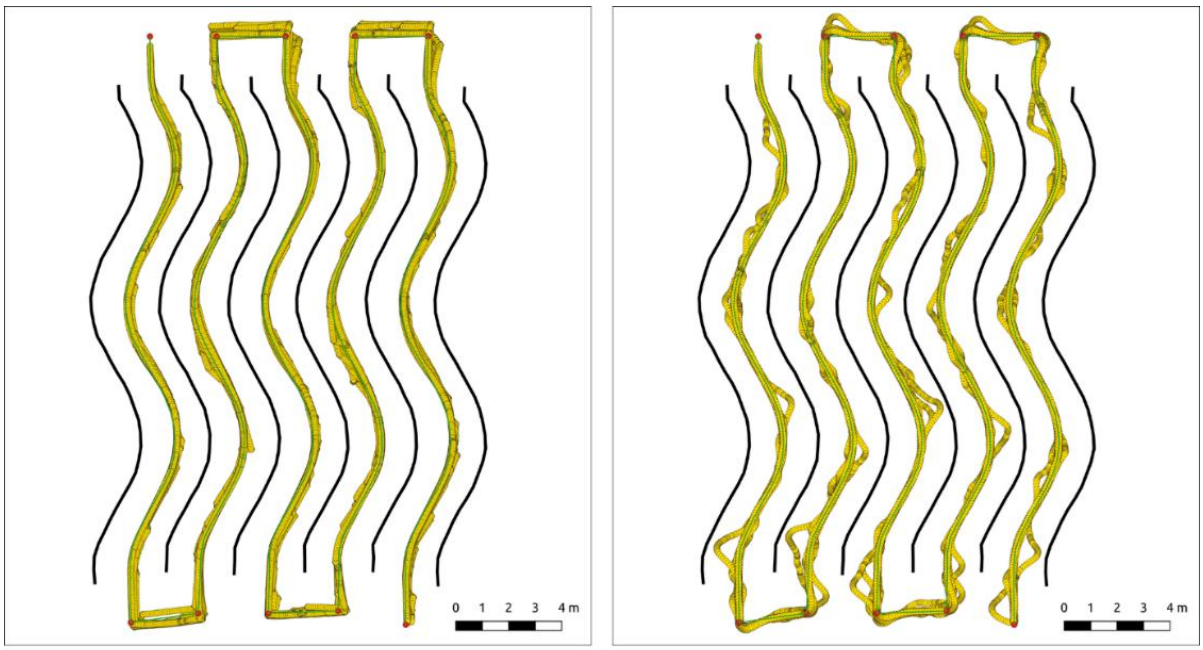
Dentro de cada combinação de área e métrica avaliada, médias seguidas da mesma letra não diferem pelo teste de Mann–Whitney U ($p < 0,05$).

Analisando-se os resultados da Tabela 3, pode-se observar que o controlador baseado em DDQN apresentou um valor médio de MAE estatisticamente menor que o PPC na área A0. Em A2 os modelos apresentaram o mesmo valor médio de MAE, não havendo diferença estatística entre eles.

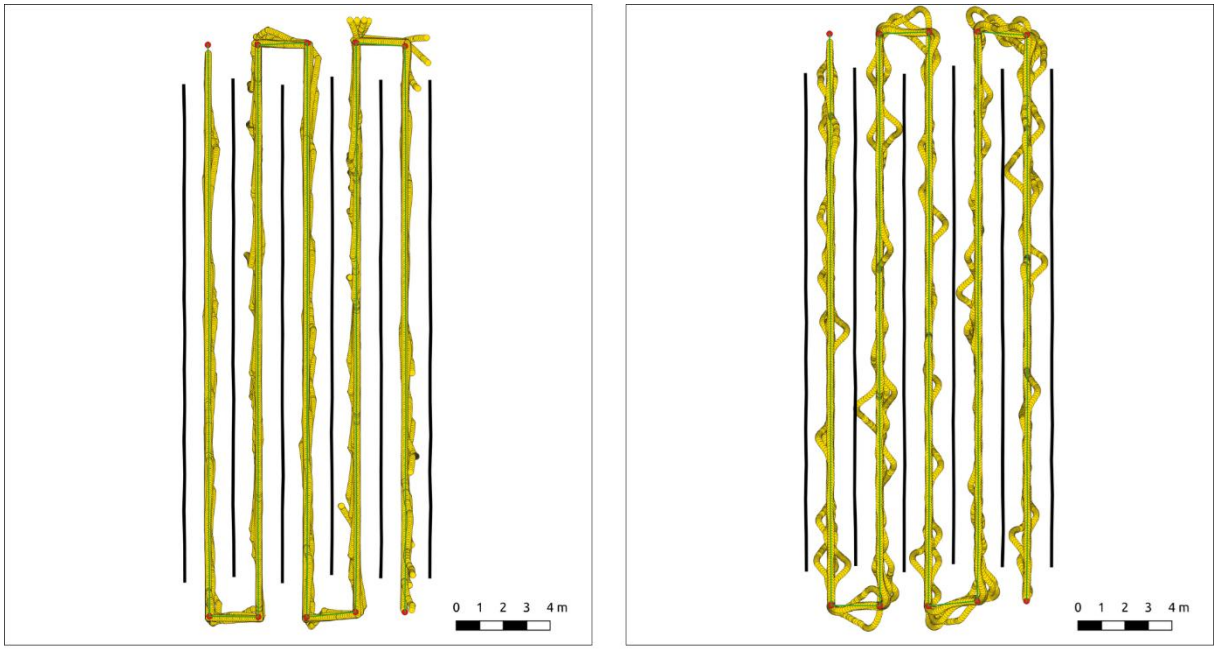
Quanto ao erro absoluto máximo, o DDQN superou o PPC nas três áreas analisadas, apresentando médias significativamente menores. Esse resultado é coerente com as análises anteriores do percentual de deslocamento sem colisão, reforçando a interpretação de que a arquitetura DDQN foi capaz de aprender uma política mais robusta frente a variações abruptas no erro do sensor GNSS, quando comparada ao PPC.

Por fim, analisando-se o RMSE do erro absoluto, encontra-se um comportamento semelhante ao observado na Tabela 2, com o controlador DDQN apresentando desempenho estatisticamente superior ao PPC em A0 e A2, mas em A1, apesar de apresentar uma média menor, a diferença não atingiu significância estatística.

Na Figura 11 são apresentadas as posições reais da plataforma robótica, obtidas antes da adição do ruído gaussiano, registradas ao longo das 30 repetições para os modelos DDQN e PPC, nas duas áreas de avaliação (A1 e A2) em cenário de falhas de GNSS. Nesta Figura estão apresentadas as repetições realizadas sem o modelo físico de colisão das plantas.



(a) (b)



(c) (d)

Figura 11 – Posições reais da plataforma robótica ao longo de 30 repetições nas áreas de avaliação com o controlador DDQN (a) e (c) e com o controlador PPC (b) e (d).
Legenda: DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.
Os pontos de posição da Plataforma robótica são apresentados em amarelo. Em preto, tem-se as linhas de plantio. As linhas de referência de deslocamento são apresentadas em verde.

Os pontos apresentados na Figura 11 mostram, de forma visual, a maior uniformidade nos resultados do DDQN e a maior capacidade que este controlador teve de atenuar desvios. Em situações onde a detecção via sensor apontaria colisões — considerando objetos a menos de 0,75 m do centro da plataforma e um espaçamento entre linhas de 2,0 m, distribuído simetricamente entre os lados da entrelinha — a margem de desvio admissível seria de 0,25 m. Nas trajetórias controladas pelo DDQN, observam-se alguns desvios que, na presença do modelo físico, configurariam colisões; contudo, tais desvios são mais acentuados no deslocamento com o controlador PPC.

Na Tabela 4 estão apresentados as médias e os desvios padrão do tempo, em segundos, necessário para que a plataforma robótica se deslocasse do ponto inicial ao ponto final, nos doze cenários experimentais, nas repetições realizadas sem os modelos físicos de colisão com as plantas. As letras de significância indicam diferenças estatísticas segundo o teste de Mann-Whitney U ($p < 0,05$).

Tabela 4 - Tempos médios e desvios padrão, em segundos, necessário para percorrer toda a área com os controladores DDQN e PPC nos cenários de deslocamento de alta precisão (sem falhas no sistema GNSS) e com falhas no sistema GNSS, para as três áreas avaliadas (A0, A1 e A2).

	Sem falhas no GNSS		Com falhas no GNSS	
	DDQN	PPC	DDQN	PPC
A0	444,9 ± 4,7 ***a***	400,4 ± 1,7 ***b***	548,1 ± 28,0 ***c***	449,1 ± 26,5 ***d***
A1	322,4 ± 1,6 ***a***	277,0 ± 1,2 ***b***	403,4 ± 65,0 ***c***	295,9 ± 10,7 ***d***
A2	291,7 ± 5,9 ***a***	282,4 ± 4,1 ***b***	311,5 ± 12,1 ***c***	321,0 ± 15,4 ***d***

Legenda: DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.
Dentro de cada combinação de área e cenário (com ou sem falhas), médias seguidas da mesma letra não diferem pelo teste de Mann–Whitney U ($p < 0,05$).

A partir dos dados apresentados na Tabela 4, observa-se que o PPC completou o percurso nas três áreas em menos tempo que o DDQN no cenário sem falhas no sistema GNSS. Acredita-se que esta diferença de desempenho está associada à discretização das velocidades adotada para adaptação do DDQN, que limitou a rotação máxima das rodas a 3 rad/s e restringiu a diferença de velocidade angular entre elas a intervalos de 10%. Um indício desta limitação é observado no fato de que na área A2 — composta predominantemente por linhas retas, onde ambos os modelos tendem a manter velocidades próximas à base durante a maior parte do trajeto, ocorreu a menor diferença de tempo entre os dois modelos.

A Tabela 5 apresenta as variações nas médias do tempo necessário para a conclusão do percurso por ambos os modelos, nas três áreas, no cenário com falha de GNSS, em comparação ao cenário sem falhas.

Tabela 5 - Variações nas médias do tempo necessário para a conclusão do percurso por ambos os modelos, nas três áreas, no cenário com falha no sensor GNSS, em comparação ao cenário sem falhas.

	DDQN	PPC
A0	+23,2%	+12,2%
A1	+25,1%	+6,8%
A2	+6,8%	+13,7%

Legenda: DDQN = Double Deep Q-Network; PPC = controlador Pure Pursuit.

Os dados da Tabela 5 representam o impacto das falhas do sistema GNSS no tempo de conclusão do percurso, evidenciando diferenças fundamentais nas estratégias de robustez dos controladores. Os resultados indicam que o agente DDQN desenvolveu uma política de controle adaptativa e sensível ao contexto operacional, em contraste com o comportamento predominantemente reativo e uniforme do PPC.

Nas áreas com maior complexidade de manobras (A0 e A1), o aumento percentual no tempo de deslocamento do DDQN foi superior ao observado para o PPC. Acredita-se que tal comportamento reflete a adoção de uma estratégia deliberada de mitigação de risco. Diante da elevada incerteza causada por falhas de do sensor GNSS em curvas, o agente de RL reduziu intencionalmente sua velocidade de avanço, priorizando a estabilidade da trajetória e a prevenção de falhas críticas, ainda que em detrimento da velocidade de execução.

A eficácia dessa política é reforçada pelos resultados obtidos na Área Reta (A2), onde o padrão se inverte. Nesse cenário, o DDQN apresentou menor impacto no tempo de deslocamento, com um aumento de apenas 6,8%, enquanto o PPC registrou acréscimo de 13,7%. Esse desempenho sugere que o agente aprendeu a identificar e atenuar perturbações puramente laterais em trajetórias retilíneas, evitando as correções excessivas que, devido à sua natureza geométrica, o PPC inevitavelmente executa.

De forma geral, a análise do tempo de deslocamento reforça não apenas a maior robustez do DDQN, mas também fornece evidências de que ele aprendeu uma política de controle sofisticada, capaz de alternar entre comportamentos cautelosos em cenários de alto risco (curvas) e respostas estáveis e seletivas a perturbações em situações de baixo risco (retas).

5. Conclusões

Neste trabalho foi desenvolvido um ambiente de simulação customizável, construído sobre ROS 2 e Gazebo, com capacidades de geração paramétrica de lavouras e modelagem de incertezas, incluindo ruído no sistema GNSS, falhas de sinal intermitentes e patinagem de rodas. Os resultados evidenciam que o algoritmo proposto constitui uma solução eficiente, ágil e flexível para a criação de ambientes virtuais destinados ao treinamento e à validação de modelos de controle autônomo em sistemas robóticos agrícolas. Tal capacidade não apenas viabiliza experimentos controlados e reprodutíveis, como também reduz custos e riscos associados a testes em campo, acelerando o ciclo de desenvolvimento de tecnologias para veículos agrícolas autônomos.

Nos testes de navegação conduzidos, os resultados obtidos demonstram que, em condições ideais de operação, ambos os controladores — PPC e DDQN — foram capazes de garantir a navegação segura e precisa, atingindo 100% de trajetórias sem colisões e erros compatíveis com os reportados na literatura. Entretanto, a introdução de falhas bruscas no sinal de GNSS revelou diferenças significativas de desempenho. O agente baseado em Deep Double Q-Network apresentou maior resiliência, mantendo menor degradação das métricas de erro e preservando um percentual superior de área percorrida sem colisões em todos os cenários avaliados. Em contrapartida, o PPC, apesar de eficiente em condições normais, mostrou-se mais sensível a distúrbios, o que reduziu substancialmente sua capacidade de manter a trajetória planejada sob falhas.

As análises realizadas indicam que, para a navegação robusta em cenários agrícolas realistas, abordagens baseadas em aprendizado por reforço profundo oferecem vantagens substanciais em termos de segurança, estabilidade e confiabilidade, quando comparadas a métodos geométricos clássicos. A capacidade do agente DDQN de aprender políticas de controle não lineares e adaptativas a partir da experiência demonstrou-se eficaz para lidar com incertezas severas inerentes à operação no campo.

A principal limitação deste estudo reside no fato de ter sido conduzido exclusivamente em ambiente de simulação. Pesquisas futuras devem priorizar a validação da metodologia em plataformas robóticas reais, explorando estratégias de transferência de conhecimento entre o simulador e o ambiente físico (sim-to-real).

Além disso, o framework desenvolvido possibilita desdobramentos relevantes, como a ampliação do espaço de estados do agente para incluir dados de percepção em tempo real (por exemplo, LIDAR), favorecendo a navegação reativa e o desvio de obstáculos não mapeados, bem como a investigação de algoritmos de aprendizado por reforço mais recentes, voltados para controle contínuo.

6. Referências

Alibabaei, K.; Gaspar, P. D.; Assunção, E.; Alirezazadeh, S.; Lima, T. M.; Soares, V. N. G. J.; Caldeira, J. M. L. P. Comparison of On-Policy Deep Reinforcement Learning A2C with Off-Policy DQN in Irrigation Optimization: A Case Study at a Site in Portugal. *Computers*, v. 11, n. 7, 2022.

Arad, B.; Balendonck, J.; Barth, R.; Ben-Shahar, O.; Edan, Y.; Hellström, T.; Hemming, J.; Kurtser, P.; Ringdahl, O.; Tielen, T.; Tuijl, B. Van. Development of a sweet pepper harvesting robot. Journal of Field Robotics, v. 37, n. 6, p. 1027–1039, 2020.

Arunachalam, A.; Andreasson, H. Real-time plant phenomics under robotic farming setup: A vision-based platform for complex plant phenotyping tasks. *Computers and Electrical Engineering*, v. 92, 2021.

Atefi, A. et al. Robotic Technologies for High-Throughput Plant Phenotyping: Contemporary Reviews and Future Perspectives. *Frontiers in Plant Science*, v. 12, 2021.

Calvert, B.; Olsen A.; Whinney J.; Rahimi, A.M. Robotic Spot Spraying of Harrisia Cactus (*Harrisia martinii*) in Grazing Pastures of the Australian Rangelands. *Plants*., vol. 10, n.10, p.2054, 2021.

Chen, Y. J.; Jhong, B. G.; Chen, M. Y. A Real-Time Path Planning Algorithm Based on the Markov Decision Process in a Dynamic Environment for Wheeled Mobile Robots. *Actuators*, v. 12, n. 4, 2023.

Coulter, R. C. Implementation of the pure pursuit path tracking algorithm. 1992.

Cubero, S.; Marco-Noales, E.; Aleixos, N.; Barbé, S.; Blasco, J. Robhortic: A field robot to detect pests and diseases in horticultural crops by proximal sensing. *Agriculture* (Switzerland), v. 10, n. 7, p. 1–13, 2020.

Fernando, S.; Nethmi, R.; Silva, A.; Perera, A.; Silva, R. De; Abeygunawardhana, P. K. W. Intelligent disease detection system for greenhouse with a robotic monitoring system. ICAC 2020 - 2nd International Conference on Advancements in Computing, *Proceedings*, p. 204–209, 2020.

Fountas, S.; Mylonas, N.; Malounas, I.; Rodias, E.; Santos, C. H.; Pekkeriet, E. Agricultural robotics for field operations. *Sensors* (Switzerland), v. 20, n. 9, 2020.

Gongal, A.; Amatya, S.; Karkee, M.; Zhang, Q.; Lewis, K. Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, v. 116, p. 8–19, 2015.

Jin, X.; Mccullough, P. E.; Liu, T.; Yang, D.; Zhu, W.; Chen, Y.; Yu, J. A smart sprayer for weed control in bermudagrass turf based on the herbicide weed control spectrum. *Crop Protection*, v. 170, 2023.

Koenig, N.,Howard, A. (n.d.). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2149–2154. https://doi.org/10.1109/IROS.2004.1389727.

Kuznetsova A., Maleva T, Soloviev V. Using YOLOv3 Algorithm with Pre- and Post-Processing for Apple Detection in Fruit-Harvesting Robot. *Agronomy*, v. 10, n.7, 2020.

Loukatos, D.; Templalexis, C.; Lentzou, D.; Xanthopoulos, G.; Arvanitis, K. G. Enhancing a flexible robotic spraying platform for distant plant inspection via high-quality thermal imagery data. *Computers and Electronics in Agriculture*, v. 190, 2021.

Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics, 7(66). https://doi.org/10.1126/scirobotics.abm6074.

Midtiby, H. S.; Åstrand, B.; Jørgensen, O.; Jørgensen, R. N. Upper limit for context–based crop classification in robotic weeding applications. *Biosystems Engineering*, v. 146, p. 183–192, 2016.

Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*, v. 518, n. 7540, p. 529–533, 2015.

Pradel, M.; Fays, M.D.; Seguineau, C. Comparative life cycle assessment of intra-row and inter-row weeding practices using autonomous robot systems in French vineyards. *Science of the total environment,* v. 838, 2022.

Sun, C.; Van Der Tol, R.; Melenhorst, R.; Pacheco, L. A. P.; Koerkamp, P. G. (2024). Path planning of manure-robot cleaners using grid-based reinforcement learning. *Computers and Electronics in Agriculture*, *226*, 109456. https://doi.org/10.1016/j.compag.2024.109456.

Sutton, R. S.; Barto, A. G. Reinforcement learning: An introduction. second edition. London, England: The MIT Press, 2018.

Thrun, S.; Schwartz, A. Issues in Using Function Approximation for Reinforcement Learning. Proceedings of the 4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum, p. 1–9, 1993.

Vahdanjoo, M.; Gislum, R.; Sørensen, C. A. G. Operational, Economic, and Environmental Assessment of an Agricultural Robot in Seeding and Weeding Operations. *AgriEngineering*, v. 5, n. 1, p. 299–324, 2023.

Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-Learning. 30th AAAI Conference on Artificial Intelligence, AAAI 2016, p. 2094–2100, 2016.

Vougioukas, S. G. Agricultural robotics. Annual review of control, robotics, and autonomous systems, v. 2, p. 365-392, 2019.

Wang, Z.; Hong, T. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy,* v. 269, 2020.

Wenyu Zhang, Jingyao Gai, Zhigang Zhang, Lie Tang, Qingxi Liao, Youchun Ding, Double-DQN based path smoothing and tracking control method for robotic vehicle navigation, Computers *and Electronics in Agriculture, v.166, 2019.*

Xu, R.; Li, C. A Review of High-Throughput Field Phenotyping Systems: Focusing on Ground Robots. *Plant Phenomics*, 2022.

Yang W., Gong C., Luo X., Zhong Y., Cui E., Hu J., Song S., Xie H., Chen W. Robotic Path Planning for Rice Seeding in Hilly Terraced Fields. *Agronomy*, v. 13, n.2, 2023.

Yang, Y. *et al*. An optimal goal point determination algorithm for automatic navigation of agricultural machinery: Improving the tracking accuracy of the Pure Pursuit algorithm. Computers and Electronics in Agriculture, v. 194, p. 106760, 2022a.

Yang, J.; NI, J., LI; Y., Wen, J.; Chen, D. (2022b). The Intelligent Path Planning System of Agricultural Robot via Reinforcement Learning. *Sensors*, *22*(12), 4316. https://doi.org/10.3390/s22124316.

Yao, Q.; Tian, Y.; Wang, Q.; Wang, S. (2020). Control Strategies on Path Tracking for Autonomous Vehicle: State of the Art and Future Challenges. IEEE Access, 8, 161211–161222. https://doi.org/10.1109/ACCESS.2020.3020075.

Yue Y.; Yufei L.; Jichun W., Noboru N., Yong He, Obstacle avoidance method based on double DQN for agricultural robots, *Computers and Electronics in Agriculture*, v. 204, 2023.

Zhang, L., Zhang, R., Zhang, D., YI, T., Ding, C.; Chen, L. (2025). Path curvature incorporated reinforcement learning method for accurate path tracking of agricultural vehicles. *Computers and Electronics in Agriculture*, *234*, 110243. https://doi.org/10.1016/j.compag.2025.110243.

Zhang, Zhibin *et al*. An adaptive vision navigation algorithm in agricultural IoT system for smart agricultural robots. Comput. Mater. Contin, v. 66, n. 1, p. 1043-1056, 2021.