UNIVERSIDADE FEDERAL DE UBERLÂNDIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
FACULDADE DE ENGENHARIA ELÉTRICA

Fabio Romero de Souza Júnior

COMPARAÇÃO ENTRE ALGORITMOS DE APRENDIZADO POR REFORÇO PROFUNDO E CONTROLADOR PID APLICADOS AO SEGUIMENTO DE LINHA

UBERLÂNDIA
2026

Fabio Romero de Souza Júnior

COMPARAÇÃO ENTRE ALGORITMOS DE APRENDIZADO POR REFORÇO PROFUNDO E CONTROLADOR PID APLICADOS AO SEGUIMENTO DE LINHA

Dissertação apresentada ao programa de Pós-graduação da Faculdade de Engenharia Elétrica da Universidade federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciências

Área de concentração: Processamento Digital de Sinais e Redes de Comunicação

Orientador: Daniel Costa Ramos

UBERLÂNDIA

2026

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S729 Souza Júnior, Fabio Romero de, 1997-
2026 Comparação entre Algoritmos de Aprendizado por Reforço Profundo e Controlador PID Aplicados ao Seguimento de Linha [recurso eletrônico] / Fabio Romero de Souza Júnior. - 2026.

Orientador: Daniel Costa Ramos.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Elétrica.
Modo de acesso: Internet.
DOI http://doi.org/10.14393/ufu.di.2026.694
Inclui bibliografia.

1. Engenharia elétrica. I. Ramos, Daniel Costa,1984-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Fabio Romero de Souza Júnior

COMPARAÇÃO ENTRE ALGORITMOS DE APRENDIZADO POR REFORÇO PROFUNDO E CONTROLADOR PID APLICADOS AO SEGUIMENTO DE LINHA

Dissertação apresentada ao programa de Pós-graduação da Faculdade de Engenharia Elétrica da Universidade federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciências

Área de concentração: Processamento digital de sinais e redes de comunicação

Uberlândia,31 de agosto de 2026

Banca Examinadora:

_____

Nome – Titulação (sigla da instituição)

_____

Nome – Titulação (sigla da instituição)

_____

Nome – Titulação (sigla da instituição)

_____

Nome – Titulação (sigla da instituição)

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica
Av. João Naves de Ávila, 2121, Bloco 3N - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4707 - www.posgrad.feelt.ufu.br - copel@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Engenharia Elétrica				
Defesa de:	Dissertação de Mestrado, 829, PPGEELT				
Data:	Trinta e um de agosto de dois mil e vinte e seis	Hora de início:	14:00	Hora de encerramento:	16:45
Matrícula do Discente:	12412EEL003				
Nome do Discente:	Fabio Romero de Souza Junior				
Título do Trabalho:	Comparação entre Algoritmos de Aprendizado por Reforço Profundo e Controlador PID Aplicados ao Seguimento de Linha				
Área de concentração:	Processamento da Informação				
Linha de pesquisa:	Processamento Digital de Sinais e Redes de Comunicação				
Projeto de Pesquisa de vinculação:	Coordenador do projeto: Daniel Costa Ramos. Título do Projeto: Divulgação e Comunicação Científica de Robótica em Patos de Minas. Agência financiadora: FAPEMIG. Número do processo na agência financiadora: APQ-02616-22. Vigência do projeto: dez/2022-abril/2026.				

Reuniu-se através de videoconferência, a Banca Examinadora designada pelo Colegiado do Programa de Pós-graduação em Engenharia Elétrica, assim composta:

Doutores: Gabriela Vieira Lima (UFU), Sidney Roberto Dias de Carvalho (SLB) e Daniel Costa Ramos, orientador do discente.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Daniel Costa Ramos, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

APROVADO.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre. O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme, foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Daniel Costa Ramos**, **Professor(a) do Magistério Superior**, em 31/08/2026, às 16:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gabriela Vieira Lima**, **Professor(a) do Magistério Superior**, em 31/08/2026, às 16:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **SIDNEY ROBERTO DIAS DE CARVALHO**, **Usuário Externo**, em 31/08/2026, às 16:54, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **7647353** e o código CRC **305DBD91**.

Referência: Processo nº 23117.060888/2026-75 SEI nº 7647353

Dedico este trabalho aos meus pais, minha irmã e aos amigos que tornaram essa jornada possível.

AGRADECIMENTOS

Agradeço aos meus pais, Fábio e Erlan, e à minha irmã, Kamilla, pelo apoio, carinho e incentivo ao longo dessa caminhada. Obrigado por sempre acreditarem em mim e por não me deixarem desistir, mesmo nos momentos mais difíceis.

Aos meus amigos Dr. Eduardo Tavares, Dr. Gabriel Gonçalves, Dra. Andréia Coelho, Dr. Leandro Duarte, MSc. Gabriel Machado e MSc. Rafael Sousa, agradeço pela amizade, pelas conversas, pelos conselhos e por todo o apoio durante essa jornada.

Ao MSc. Gustavo Domingos Coelho, deixo um agradecimento especial. Sua presença ao longo de todos os dias desta jornada, sua paciência, dedicação e amizade foram fundamentais para que eu chegasse até aqui.

Ao MSc. Bruno Damasceno, agradeço pelo suporte e pela disposição em ajudar ao longo desse caminho.

À Cínara, ao Caio e à Andressa, agradeço pela amizade, pelo carinho e pelo apoio ao longo da pós-graduação. Em especial à Cínara, por todo o cuidado e acolhimento durante essa caminhada.

Por fim, ao meu orientador, Prof. Dr. Daniel Costa Ramos, meu mais profundo agradecimento. Sua orientação, confiança e apoio foram fundamentais para a realização deste trabalho. Sou muito grato por todos os ensinamentos e pela oportunidade de ter sido seu orientando.

Agradeço ao Programa de Pós-Graduação em Engenharia Elétrica (PPG-EELT) da Faculdade de Engenharia Elétrica (FEELT) da Universidade Federal de Uberlândia (UFU) pelo suporte institucional, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, pelo apoio financeiro, e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) pelo apoio ao Projeto RoboPatos (APQ-02616-22).

“A ciência é uma tentativa, em grande parte
bem-sucedida, de entender o mundo.”
(SAGAN, Carl)

RESUMO

Este trabalho apresenta uma avaliação experimental comparativa entre seis algoritmos de aprendizado por reforço profundo *Deep Deterministic Policy Gradient* (DDPG), *Twin Delayed* DDPG (TD3), *Soft Actor-Critic* (SAC), *Proximal Policy Optimization* (PPO), *Advantage Actor-Critic* (A2C) e Model-Based *Reinforcement Learning* (MBRL) e o controlador Proporcional-Integral-Derivativo (PID), aplicados ao problema de seguimento de linha por visão computacional em um robô móvel diferencial. Os experimentos foram conduzidos em ambiente simulado, considerando três cenários de complexidade geométrica crescente e duas velocidades de operação, correspondentes a 50% e 75% da velocidade máxima estável do robô. A avaliação considerou métricas de precisão, estabilidade, esforço de controle, taxa de conclusão e robustez. Os resultados indicam que o DDPG apresentou o melhor equilíbrio entre precisão e taxa de conclusão nos cenários avaliados, mantendo desempenho consistente mesmo com o aumento da velocidade. O PID demonstrou resultados competitivos em condições estáveis, com menor esforço de controle, mas apresentou sensibilidade ao aumento de velocidade. Os demais algoritmos de aprendizado por reforço apresentaram erros de rastreamento significativamente maiores. Os resultados reforçam o potencial do aprendizado por reforço profundo para tarefas de controle contínuo em robótica móvel, evidenciando que a escolha do método mais adequado depende dos requisitos específicos da aplicação.

Palavras-chave: Aprendizado por Reforço Profundo. Seguimento de linha. Visão Computacional. Robótica Móvel. Controlador PID.

ABSTRACT

This work presents an experimental comparative evaluation of six deep reinforcement learning algorithms Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Model-Based Reinforcement Learning (MBRL) and a Proportional-Integral-Derivative (PID) controller, applied to the line-following task using computer vision on a differential mobile robot. The experiments were conducted in a simulated environment, considering three scenarios of increasing geometric complexity and two operating speeds, corresponding to 50% and 75% of the robot's maximum stable velocity. The evaluation considered metrics of precision, stability, control effort, completion rate, and robustness. The results indicate that DDPG achieved the best balance between accuracy and completion rate across all evaluated scenarios, maintaining consistent performance even as speed increased. The PID controller demonstrated competitive results under stable conditions, with lower control effort, but showed sensitivity to speed increases. The remaining *reinforcement learning* algorithms presented significantly higher tracking errors. Overall, the results reinforce the potential of deep *reinforcement learning* for continuous control tasks in mobile robotics, highlighting that the most suitable method depends on the specific requirements of the application.

Keywords: *Deep Reinforcement Learning. Trajectory Following. Computer Vision. Mobile Robotics. PID Controller.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Termo em inglês	Tradução em português
A2C	Advantage Actor-Critic	Ator-Crítico com Vantagem
DDPG	Deep Deterministic Policy Gradient	Gradiente de Política Determinística Profunda
DQN	Deep Q-Network	Rede Q Profunda
DRL	Deep Reinforcement Learning	Aprendizado por Reforço Profundo
IMU	Inertial Measurement Unit	Unidade de Medição Inercial
MAE	Mean Absolute Error	Erro Absoluto Médio
MBRL	Model-Based Reinforcement Learning	Aprendizado por Reforço Baseado em Modelo
MDP	Markov Decision Process	Processo de Decisão de Markov
PID	Proportional-Integral-Derivative	Proporcional-Integral-Derivativo
PPO	Proximal Policy Optimization	Otimização de Política Proximal
RI	Robustness Index	Índice de Robustez
RL	Reinforcement Learning	Aprendizado por Reforço
RMSE	Root Mean Square Error	Raiz do Erro Quadrático Médio
ROI	Region of Interest	Região de Interesse
ROS	Robot Operating System	Sistema Operacional para Robôs
SAC	Soft Actor-Critic	Ator-Crítico Suave
SDF	Simulation Description Format	Formato de Descrição de Simulação
TD3	Twin Delayed Deep Deterministic Policy Gradient	Gradiente de Política Determinística Profunda com Atraso Duplo

SUMÁRIO

1 INTRODUÇÃO

A robótica móvel engloba uma ampla variedade de aplicações, desde a logística industrial até a exploração autônoma, nas quais o desempenho do sistema depende diretamente da qualidade e robustez do controle de movimento (Solanes e Gracia, 2025). Nesse contexto, o seguimento de linha constitui um problema fundamental, amplamente empregado em ambientes industriais estruturados para validação de estratégias de controle e percepção, servindo ainda como base para sistemas de navegação autônoma (Brancalião *et al.*, 2022; Brigido e Parente de Oliveira, 2025). O principal desafio consiste em manter o robô sobre a trajetória com desvio mínimo, mesmo na presença de incertezas e variações dinâmicas (Carlucho, Paula, De e Acosta, 2019).

Tradicionalmente, essa tarefa é tratada por meio de controladores clássicos, como o controlador Proporcional-Integral-Derivativo (*Proportional-Integral-Derivative*, PID), amplamente utilizado devido à sua simplicidade e eficácia em diversas aplicações robóticas (Yu *et al.*, 2021; Zhang, 2024). No entanto, seu desempenho está intrinsecamente associado a condições estáveis de operação (Wang *et al.*, 2022). Em ambientes dinâmicos, sua eficácia torna-se limitada, com degradação de desempenho na presença de variações de carga, perturbações externas e incertezas de modelo, exigindo frequentemente a re-sintonia manual dos ganhos e comprometendo a robustez do sistema (Leal *et al.*, 2025; Mohindru, 2024).

Antes da consolidação de abordagens baseadas em redes neurais profundas, técnicas de aprendizado por reforço já eram empregadas no controle de robôs móveis, inclusive em conjunto com controladores clássicos. (Carlucho, Paula, De e Acosta, 2019), por exemplo, propuseram o algoritmo Double Q-PID, no qual um agente baseado em *Q-learning* ajusta dinamicamente os parâmetros de um controlador PID a partir da interação com o ambiente, reduzindo a dependência de uma sintonia manual. Entretanto, métodos tabulares como o *Q-learning* requerem a discretização dos espaços de estados e ações, o que pode limitar a resolução das decisões de controle e sua aplicação a problemas de maior complexidade. Diante dessa limitação, o aprendizado por reforço profundo (*Deep Reinforcement Learning*, DRL) surge como uma evolução dessas abordagens, empregando redes neurais para aproximar funções de valor ou políticas e permitindo trabalhar com representações de maior dimensionalidade (Arulkumaran *et al.*, 2017).

Desde então, o uso de algoritmos de aprendizado por reforço profundo tem se expandido, oferecendo a capacidade de aprender políticas de controle por meio da interação

direta com o ambiente (Liu *et al.*, 2022; Tang *et al.*, 2025). Apesar do crescente uso de DRL na robótica móvel (Park, Jun e Lee, 2025), a literatura ainda carece de avaliações sistemáticas que comparem simultaneamente múltiplos paradigmas de aprendizado sob diferentes condições de operação. A maioria dos estudos limita-se a comparar um único algoritmo DRL contra o PID, desconsiderando nuances entre métodos baseados em modelo e sem modelo, bem como abordagens *on-policy* e *off-policy*, quando submetidos a variações de velocidade e complexidade ambiental sob métricas unificadas (ZHU, Y. et al.., 2025).

Para viabilizar avaliações sistemáticas desse tipo, a utilização de ambientes simulados desempenha papel relevante no desenvolvimento de estratégias de controle baseadas em aprendizado por reforço, uma vez que o treinamento diretamente em plataformas físicas pode envolver custos elevados, longos períodos de execução e riscos de danos ao robô. Nesse sentido, (Pereira *et al.*, 2026) identificaram que 16 dos 19 estudos analisados em sua revisão sistemática empregaram ambientes de simulação, permitindo testar algoritmos de forma segura, acelerar o processo de treinamento e explorar cenários de maior complexidade. Essa abordagem também favorece a realização de experimentos controlados e repetíveis, importantes para a comparação entre diferentes estratégias de controle.

Diante dessa necessidade de avaliações controladas e comparáveis, este trabalho apresenta uma avaliação experimental comparativa entre seis algoritmos de DRL: Gradiente de Política Determinística Profunda (*Deep Deterministic Policy Gradient*, DDPG), Gradiente de Política Determinística Profunda com Atraso Duplo (*Twin Delayed Deep Deterministic Policy Gradient*, TD3), Crítico de Ator com Entropia Suave *(Soft Actor-Critic,* SAC), Otimização de Política Proximal (*Proximal Policy Optimization*, PPO), Crítico de Ator com Vantagem (*Advantage Actor-Critic*, A2C) e Aprendizado por Reforço Baseado em Modelo *(Model-Based Reinforcement Learning*, MBRL), e o controlador PID, aplicados ao seguimento de linha por visão computacional em ambiente simulado. Os controladores são avaliados em três cenários de complexidade geométrica crescente e duas velocidades de operação.

O documento está organizado da seguinte forma: o Capítulo 2 apresenta o referencial teórico, abordando os conceitos de robótica móvel, controle clássico e aprendizado por reforço profundo. O Capítulo 3 descreve a metodologia adotada, incluindo o ambiente experimental, os controladores avaliados e o protocolo experimental. O Capítulo 4 apresenta o desenvolvimento da plataforma experimental e a implementação dos controladores. O Capítulo 5 apresenta e discute os resultados obtidos. Por fim, a Conclusão sintetiza os principais resultados da pesquisa, suas limitações e as perspectivas para trabalhos futuros.

1.1 Objetivo Geral

Este trabalho avalia comparativamente controladores clássicos e algoritmos de aprendizado por reforço aplicados ao seguimento de linha por visão computacional em robôs móveis, investigando o desempenho, a robustez e as condições de aplicabilidade de cada abordagem em cenários simulados.

1.2 Objetivo Específico

• Implementar um ambiente de simulação para a tarefa de seguimento de linha, permitindo a avaliação controlada dos diferentes métodos de controle.

• Desenvolver e treinar os algoritmos DDPG, TD3, SAC, PPO, A2C e MBRL para o seguimento de linha, considerando diferentes configurações de mapas e velocidades de operação.

• Definir e aplicar métricas de precisão, estabilidade e esforço de controle, possibilitando uma análise quantitativa entre os métodos clássicos e os algoritmos de DRL.

• Realizar uma comparação sistemática entre todas as abordagens, identificando pontos fortes, limitações e condições em que cada técnica apresenta melhor desempenho.

• Oferecer uma base experimental que contribua para o desenvolvimento de estratégias de controle mais adaptativas e robustas para robôs seguidores de linha.

1.3 Justificativa

O seguimento de linha por visão computacional em robôs móveis constitui uma tarefa fundamental para a validação de estratégias de navegação autônoma, sendo amplamente empregado em aplicações como robôs de serviço, veículos autônomos e plataformas móveis (Brancalião *et al.*, 2022; Brigido e Parente de Oliveira, 2025). Tradicionalmente, esse problema é tratado por controladores clássicos, como o PID, devido à sua simplicidade de implementação e baixo custo computacional. Entretanto, esses controladores dependem de sintonia manual e apresentam redução de desempenho diante de mudanças nas condições de operação, como diferentes velocidades e geometrias de trajetória (Leal et al., 2025; Mohindru, 2024).

O *Reinforcement Learning* (RL) tem sido investigado como uma alternativa para superar essas limitações, permitindo que agentes aprendam políticas de controle por interação com o

ambiente, sem a necessidade de modelagem explícita da dinâmica do sistema (Tang et al., 2025). Nesse contexto, diferentes algoritmos, como DDPG, TD3, SAC, PPO, A2C e abordagens *model-based*, vêm sendo empregados em tarefas de navegação autônoma e controle contínuo, demonstrando resultados promissores (Liu, 2024; Park, Jun e Lee, 2025).

Entretanto, os estudos de revisão disponíveis evidenciam que essa evolução ainda não se reflete no problema específico de seguimento de linha por visão computacional. A revisão sistemática (Pereira *et al.*, 2026) identificou um número reduzido de trabalhos que aplicam RL a essa tarefa, predominando abordagens baseadas em *Q-learning* tabular ou na utilização isolada de um único algoritmo. De forma complementar, a revisão da literatura sobre navegação autônoma (Brigido e Parente de Oliveira, 2025) mostra que, embora diversos algoritmos de RL tenham sido empregados em tarefas gerais de navegação, as avaliações concentram-se em problemas distintos do seguimento de linha por visão computacional e, em geral, não realizam comparações sistemáticas entre diferentes abordagens de RL.

As soluções propostas na literatura para o controle de robôs seguidores de linha baseadas em aprendizado permanecem fragmentadas. Os trabalhos que empregam RL concentram-se em abordagens como Q-learning tabular (Krawczyk *et al.*, 2024) ou na utilização do aprendizado para ajuste de parâmetros de controladores clássicos, como PID fuzzy (Lee e Sung, 2022). Por outro lado, métodos de seguimento de linha baseados em visão computacional têm priorizado técnicas de aprendizado supervisionado (Leal *et al.*, 2025), em vez de políticas aprendidas por interação com o ambiente. Em conjunto, esses estudos evidenciam a ausência de investigações que avaliem sistematicamente algoritmos de RL com ações contínuas aplicados diretamente ao seguimento de linha por visão computacional.

Diante desse cenário, este trabalho realiza uma comparação sistemática entre diferentes estratégias de RL, DDPG, TD3, SAC, PPO, A2C e MBRL utilizando um controlador PID como referência. A avaliação é conduzida em três cenários de complexidade crescente e duas velocidades de operação, considerando métricas de precisão de seguimento, estabilidade, esforço de controle e taxa de conclusão.

1.4 Produção científica

O desenvolvimento desta dissertação resultou na publicação de dois artigos científicos relacionados aos principais produtos desta pesquisa.

O primeiro artigo, intitulado Plataforma baseada em ROS2 e Gazebo para avaliação de controladores em robótica móvel com visão computacional e inteligência artificial, apresenta a

plataforma experimental utilizada neste trabalho, incluindo sua arquitetura e integração dos módulos de simulação, controle e visão computacional. O artigo foi publicado na revista E&S Engineering and Science (2026), DOI: 10.18607/ES20261521680.(Souza Junior, De e Costa Ramos, 2026).

O segundo artigo, intitulado Control Strategy Comparison for an Intelligent Mobile Mechanism, apresenta uma comparação entre diferentes estratégias de controle para um robô móvel inteligente em ambiente simulado, sintetizando os principais resultados experimentais desta dissertação. O trabalho foi publicado como capítulo de livro em 2026, DOI: 10.1007/978-3-032-30258-8_29. (Souza Júnior, de e Ramos, 2027).

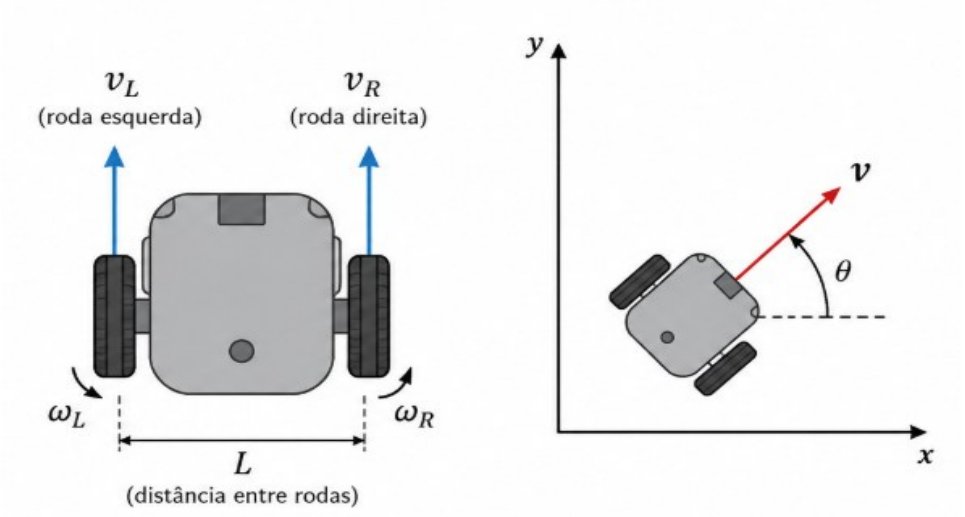
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo estabelece as bases teóricas necessárias para a compreensão do trabalho, abordando os conceitos fundamentais de robótica móvel, controle clássico, aprendizado por reforço e aprendizado por reforço profundo, bem como as ferramentas e plataformas utilizadas na pesquisa.

2.1 Robótica Móvel e Controle Clássico

Robôs diferenciais são robôs móveis caracterizados por possuírem duas rodas motorizadas independentes, cuja trajetória e direção são controladas pela diferença de velocidades entre as rodas esquerda e direita. Essa configuração permite manobras como curvas fechadas e rotação no próprio eixo. Do ponto de vista cinemático, o movimento no plano pode ser descrito por uma velocidade linear v e uma velocidade angular ω, de modo que a evolução da posição (x, y) e da orientação (θ) ao longo do tempo é dada por $\dot{x} = v \cdot \cos(\theta)$, $\dot{y} = v \cdot \text{sen}(\theta)$ e $\dot{\theta} = \omega$. Por sua vez, v e ω são obtidas a partir das velocidades angulares das rodas esquerda e direita (ωL e ωR), do raio da roda (r) e da distância entre rodas (L), segundo $v = r(\omega R + \omega L)/2$ e $\omega = r(\omega R - \omega L)/L$, conforme ilustrado na Figura 1. (Siegwart, et. al. 2011).

Figura 1 - Representação da cinemática de um robô diferencial



Fonte: o autor.

Em tarefas de seguimento de linha, diferentes estratégias de controle podem ser empregadas. Abordagens clássicas, como controladores Proporcional-Integral-Derivativo

(PID), são amplamente utilizadas devido à sua simplicidade, interpretabilidade e eficácia em diversas aplicações (Razali *et al.*, 2023). No entanto, problemas que envolvem variações de dinâmica, múltiplas condições operacionais e incertezas no ambiente têm motivado a investigação de abordagens baseadas em aprendizado, capazes de adaptar o comportamento do agente a partir da interação com o ambiente (Brancalião *et al.*, 2022).

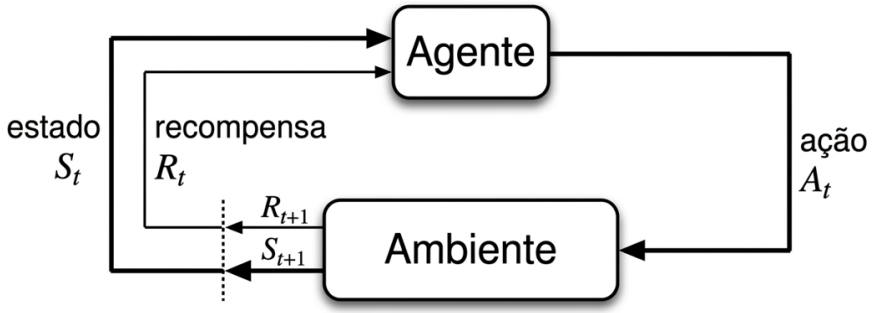
2.2 Aprendizado por Reforço (Reinforcement Learning - RL)

O Aprendizado por Reforço (*Reinforcement Learning – RL*) é uma área do aprendizado de máquina na qual um agente aprende a tomar decisões sequenciais por meio da interação com o ambiente, com o objetivo de maximizar uma recompensa acumulada ao longo do tempo. Em robótica móvel, essa abordagem permite o aprendizado de políticas de controle sem a necessidade de modelagem explícita do sistema, sendo adequada para cenários com incertezas e dinâmicas complexas, nos quais métodos tradicionais, como o controlador PID, podem apresentar limitações (Sutton e Barto, 2018).

2.2.1 Elementos do Aprendizado por Reforço

Os principais componentes de um sistema de Aprendizado por Reforço são o agente, o ambiente, os estados, as ações, as recompensas, a política e a função de valor. O agente (agent) é a entidade responsável pela tomada de decisão e pelo aprendizado, interagindo com o ambiente por meio da seleção de ações com base nas observações disponíveis. No contexto deste trabalho, o agente corresponde ao módulo de controle inteligente que governa o comportamento do robô diferencial, determinando a ação a partir do erro lateral estimado. A Figura 2 ilustra a interação entre o agente e o ambiente durante o processo de aprendizado.

Figura 2 - A interação agente-ambiente no aprendizado por reforço



Fonte: o autor.

O ambiente *(environment)* corresponde ao conjunto de elementos e condições externas com os quais o agente interage, incluindo a dinâmica do sistema, o cenário e possíveis perturbações. Em cada instante, o ambiente é representado por um estado (*state*), que reúne as variáveis relevantes para a tomada de decisão do agente. Em aplicações robóticas, o estado pode ser composto por medições sensoriais, variáveis cinemáticas e indicadores de erro associados à tarefa. No problema de seguimento de linha considerado neste trabalho, o estado é representado pelo erro lateral estimado a partir da posição do centróide da linha na imagem.

Com base no estado observado, o agente seleciona uma ação (*action*), correspondente ao conjunto de comandos aplicados ao ambiente. As ações podem pertencer a espaços discretos, contínuos ou híbridos, dependendo da natureza do problema. Neste trabalho, a ação corresponde à velocidade angular aplicada ao robô. Após a execução da ação, o ambiente realiza a transição para um novo estado, dando continuidade ao processo de interação.

Após a execução de uma ação, o ambiente fornece uma recompensa (*reward*), que corresponde ao sinal de realimentação utilizado para avaliar a qualidade da decisão tomada pelo agente. Recompensas mais elevadas estão associadas a comportamentos desejáveis, enquanto valores menores penalizam ações inadequadas, de modo que o objetivo do agente é maximizar a recompensa acumulada ao longo do processo de aprendizagem. Neste trabalho, a recompensa é definida em função do erro lateral, incentivando o agente a reduzir esse erro por meio do ajuste da velocidade angular do robô.

O comportamento do agente é definido por uma política, responsável por estabelecer o mapeamento entre estados e ações. Essa política pode ser determinística, quando associa uma única ação a cada estado, ou estocástica, quando define uma distribuição de probabilidade sobre as ações possíveis. Em algoritmos de aprendizado por reforço profundo, a política é geralmente aproximada por redes neurais, que aprendem esse mapeamento a partir das interações com o ambiente.

Considere um MDP, definido pela tupla (S, A, P, R, γ), em que S representa o espaço de estados, com $s \in S$ denotando um estado do ambiente, e A representa o espaço de ações, com $a \in A$ denotando uma ação disponível ao agente (Sutton e Barto, 2018) . Além da política, o agente utiliza uma função de valor *(value function),* responsável por estimar a recompensa acumulada esperada a partir de um estado ou de um par estado-ação. Em algoritmos de DRL, essa função é geralmente aproximada por redes neurais, permitindo lidar com espaços de estados complexos. Os principais tipos são a função valor de estado, V(s), que estima o retorno esperado ao seguir uma política a partir de um estado $s \in S$, e a função valor de ação, Q(s,a),

que estima o retorno esperado ao executar uma ação a ∈ A e, posteriormente, seguir essa política.

Além da política, o agente utiliza uma função de valor *(value function),* responsável por estimar a recompensa acumulada esperada a partir de um estado ou de um par estado-ação. Em algoritmos de DRL, essa função é geralmente aproximada por redes neurais, permitindo lidar com espaços de estados complexos. Os principais tipos são a função valor de estado, V(s), que estima o retorno esperado ao seguir uma política a partir de um estado, e a função valor de ação, Q(s,a), que estima o retorno esperado ao executar uma ação e, posteriormente, seguir essa política.

A interação entre o agente e o ambiente ocorre ao longo de um episódio (*episode*), composto por uma sequência de estados, ações e recompensas. Um episódio é encerrado quando um estado terminal é atingido ou quando é alcançado o número máximo de passos definido para o treinamento.

2.2.2 Características dos Algoritmos de Aprendizado por Reforço

O Aprendizado por Reforço é tradicionalmente modelado por meio de Processos de Decisão de Markov *(Markov Decision Processes,* MDPs*),* uma estrutura matemática utilizada para representar problemas de tomada de decisão sequencial. Um MDP é caracterizado pela propriedade de Markov, segundo a qual a dinâmica do sistema depende apenas do estado atual e da ação escolhida, sendo independente do histórico de estados anteriores. Dessa forma, o estado atual contém toda a informação necessária para a tomada de decisão e para a previsão das transições futuras.

Nesse contexto, o agente interage continuamente com o ambiente buscando maximizar a recompensa acumulada ao longo do tempo. Durante esse processo, surge um dos principais desafios do aprendizado por reforço: equilibrar exploração e explotação. A exploração consiste em experimentar novas ações com o objetivo de descobrir alternativas potencialmente mais vantajosas, enquanto a explotação utiliza o conhecimento já adquirido para selecionar as ações que apresentam maior recompensa esperada (Sutton e Barto, 2018) . Diversas estratégias podem ser empregadas para estabelecer esse equilíbrio, como a política ε-greedy, em que a melhor ação conhecida é escolhida com probabilidade $1 - \varepsilon$ e, com probabilidade ε, uma ação aleatória é selecionada. Outra abordagem é a estratégia Softmax, que atribui probabilidades de seleção às ações de acordo com seus valores estimados, favorecendo ações mais promissoras sem

eliminar completamente a exploração (Sutton e Barto, 2018) . Em problemas com espaços de ação contínuos, também é comum adicionar ruído às ações geradas pela política, promovendo pequenas variações que incentivam a exploração.

Outra classificação importante dos algoritmos de RL refere-se à forma como os dados são utilizados durante o aprendizado. Os métodos podem ser *on-policy* ou *off-policy*. Nos algoritmos *on-policy*, a política utilizada para coletar as experiências é a mesma que está sendo otimizada, fazendo com que o aprendizado ocorra a partir das decisões da política corrente (Sutton e Barto, 2018) . Em contrapartida, nos métodos *off-policy*, a política aprendida pode utilizar experiências geradas por políticas diferentes, permitindo o reaproveitamento de dados previamente coletados, como aqueles armazenados em memórias de repetição (*replay buffers*). Essa característica aumenta a eficiência do treinamento e amplia a flexibilidade dos algoritmos.

Além disso, os métodos de aprendizado por reforço também podem ser classificados quanto ao uso de um modelo explícito do ambiente. No *Model-Based Reinforcement Learning* (MBRL), o agente aprende ou dispõe de um modelo capaz de estimar a dinâmica do ambiente, possibilitando prever as consequências das ações antes de executá-las. Essa abordagem tende a apresentar maior eficiência amostral, porém seu desempenho depende diretamente da precisão do modelo aprendido, sendo suscetível a erros de modelagem (Sutton e Barto, 2018). Em contraste, *no Model-Free Reinforcement Learning*, o agente aprende diretamente a política ou a função de valor a partir da interação com o ambiente, dispensando a construção explícita de um modelo. Embora normalmente exija maior quantidade de experiências para convergir, essa abordagem evita os erros decorrentes de aproximações incorretas da dinâmica do sistema.

2.3 Deep Reinforcement Learning

O *Deep Reinforcement Learning (DRL)* é uma abordagem de aprendizado por reforço que utiliza redes neurais profundas para aproximar funções de valor e/ou políticas de decisão. Diferentemente das abordagens tabulares, que armazenam explicitamente os valores associados aos estados e ações, o DRL emprega redes neurais como aproximadores, permitindo lidar com espaços de estados de alta dimensionalidade e, em determinados algoritmos, com espaços de ação contínuos. O aprendizado ocorre a partir da interação do agente com o ambiente, por meio das experiências geradas durante a execução das ações e das recompensas recebidas, sem a necessidade de um conjunto de dados previamente rotulados (Tang et al., 2025). Entre suas principais vantagens estão a capacidade de representar relações complexas e a possibilidade de lidar com espaços de estados e ações de maior dimensionalidade. Como desvantagem, o

treinamento pode apresentar instabilidade e dificuldades de convergência, além de demandar um elevado número de interações com o ambiente e maior custo computacional devido ao uso de redes neurais profundas.

Nas subseções seguintes, são apresentados os principais algoritmos de DRL utilizados neste trabalho para o controle do robô móvel: DDPG, TD3, SAC, PPO, A2C e MBRL. Cada algoritmo é descrito quanto aos seus princípios de funcionamento, características e particularidades, de modo a fundamentar as escolhas metodológicas adotadas na comparação com o controlador PID clássico.

As arquiteturas *Actor-Critic* são amplamente utilizadas em Deep Reinforcement Learning, especialmente em problemas com espaços de ações contínuos. Essas abordagens combinam dois componentes principais (Mnih et al., 2016).O *Actor* é responsável por aprender a política, realizando o mapeamento de estados para ações e gerando diretamente a ação a ser executada. Já o *Critic* é responsável por avaliar as ações selecionadas pelo *Actor*, estimando funções de valor, como (Q(s,a)) ou (V(s)), cujo sinal é utilizado para orientar a atualização da política.

O A2C é uma extensão da arquitetura *Actor-Critic* que utiliza a função valor para orientar a atualização da política. O sinal de aprendizado é obtido ao subtrair a estimativa do valor do estado atual (V(s)) do valor esperado do par estado-ação (Q(s,a)), resultando em (A(s,a) = Q(s,a) - V(s)). Isso fornece um indicador de quanto a ação (a) foi melhor (ou pior) do que o comportamento médio esperado no estado (Mnih *et al.*, 2016). Entre suas vantagens estão a redução da variância em relação ao *Actor-Critic* básico e o fornecimento de um sinal mais informativo para atualizar a política, favorecendo maior estabilidade no treinamento. Como desvantagem, o método requer a manutenção e atualização de duas redes, *Actor e Critic*, o que aumenta a complexidade de implementação e pode dificultar o ajuste e a estabilidade do processo de treinamento.

Para problemas com espaços de ações contínuas, como o controle de velocidades angulares e lineares de um robô, algoritmos específicos de DRL são empregados. O DQN, por exemplo, é nativamente para espaços de ações discretos e, portanto, não é adequado para esta tarefa sem modificações significativas.

O Model-Based Reinforcement Learning (MBRL) é uma abordagem de aprendizado por reforço que incorpora um modelo da dinâmica do ambiente para auxiliar o processo de tomada de decisão. Diferentemente dos métodos *model-free*, que aprendem exclusivamente a partir da interação direta com o ambiente, o MBRL utiliza um modelo capaz de prever as

transições de estado e, em alguns casos, as recompensas futuras. Esse modelo pode ser conhecido previamente ou aprendido durante o treinamento, permitindo realizar simulações internas para planejar ações antes de executá-las no ambiente real (Moerland *et al.*, 2020). Entre suas principais vantagens estão a maior eficiência amostral e a redução da quantidade de interações necessárias para o aprendizado. Como desvantagem, o desempenho do método depende diretamente da precisão do modelo aprendido, de modo que erros de modelagem podem acumular-se ao longo do planejamento e comprometer a qualidade da política obtida.

O PPO é um método de gradiente de política que busca aumentar a estabilidade do treinamento ao evitar atualizações excessivamente grandes na política. A função objetivo é modificada de modo que a nova política só seja aceita quando permanecer próxima da política anterior, implementado por meio de uma função objetivo com recorte (*clipped objective*). Essa estratégia limita mudanças abruptas e reduz o risco de degradação de desempenho após uma atualização, favorecendo um aprendizado mais estável (Schulman *et al.*, 2017). Entre suas vantagens estão o treinamento tipicamente estável e robusto, devido ao caráter conservador das atualizações de política. Como desvantagem, em alguns cenários o método pode apresentar convergência mais lenta, pois as atualizações são deliberadamente limitadas para manter a política próxima da anterior.

O DDPG é um algoritmo *off-policy* desenvolvido para problemas com ações contínuas, combinando ideias de métodos baseados em valor, inspirados no *Q-learning*, e métodos de gradiente de política. Sua estrutura é do tipo *Actor-Critic*. Para aumentar a estabilidade do treinamento, o DDPG utiliza redes-alvo tanto para o *Actor* quanto para o *Critic*, atualizadas de forma lenta, e emprega um *replay buffer* para armazenar transições e amostrar experiências passadas (Lillicrap *et al.*, 2015). Entre suas vantagens está a adequação para espaços de ação contínuos e de alta dimensionalidade, mantendo boa capacidade de controle em problemas complexos. Como desvantagem, o método pode ser sensível à escolha de hiperparâmetros e, em muitos casos, depende de técnicas como normalização e ajustes cuidadosos para manter o treinamento estável.

O TD3 é um algoritmo *off-policy* para ações contínuas baseado na arquitetura *Actor-Critic* com política determinística, proposto como uma melhoria do DDPG. O método aumenta a estabilidade do treinamento ao reduzir a superestimação de (Q(s,a)) por meio do uso de dois *Critic*, além de empregar atualizações atrasadas do *Actor* e suavização da política-alvo. Assim como o DDPG, utiliza *replay buffer* e redes-alvo para favorecer um aprendizado mais estável (Fujimoto, Hoof, Van e Meger, 2018). Entre suas vantagens estão a maior estabilidade e

robustez em comparação ao DDPG. Como desvantagem, apresenta maior complexidade e sensibilidade à escolha de hiperparâmetros.

O SAC é um algoritmo *off-policy* para ações contínuas baseado em arquitetura Actor-Critic, no qual o ator aprende uma política estocástica $\pi(a|s)$. Diferentemente de abordagens determinísticas, o SAC adota o princípio de máxima entropia, incorporando um termo adicional na otimização para incentivar exploração e políticas mais robustas. O SAC busca maximizar o retorno esperado e, simultaneamente, manter alta entropia, evitando convergência prematura. Por ser *off-policy*, utiliza *replay buffer* e redes-alvo para estabilizar o treinamento e melhorar a eficiência amostral (Haarnoja et al., 2018).

2.4 Controlador Proporcional-Integral-Derivativo (PID)

O controlador Proporcional-Integral-Derivativo (PID) é uma das estratégias mais utilizadas em sistemas de controle, sendo aplicado em diferentes áreas devido à sua simplicidade, flexibilidade e capacidade de fornecer uma resposta adequada a diferentes dinâmicas de sistemas. O princípio de funcionamento do PID consiste em utilizar o erro entre o valor de referência e o valor observado do sistema para determinar uma ação de controle. A partir desse erro, o controlador combina três componentes: proporcional, integral e derivativo, cada uma contribuindo de maneira distinta para o comportamento do sistema(Ogata, 2010).

O termo proporcional (P) produz uma ação de controle diretamente proporcional ao erro instantâneo. Dessa forma, quanto maior a diferença entre o valor desejado e o valor observado, maior tende a ser a ação aplicada pelo controlador. O ganho proporcional kp determina a intensidade dessa resposta. Embora o aumento desse ganho possa reduzir o erro e tornar a resposta mais rápida, valores excessivamente elevados podem provocar oscilações e comprometer a estabilidade do sistema.

O termo integral (I) considera o acúmulo do erro ao longo do tempo. Sua principal função é reduzir ou eliminar o erro em regime permanente, uma vez que erros persistentes são acumulados e produzem uma ação de controle adicional. A intensidade dessa contribuição é determinada pelo ganho integral ki. Entretanto, um ganho integral elevado pode aumentar o tempo de resposta e favorecer oscilações ou fenômenos associados ao acúmulo excessivo do erro.

O termo derivativo (D) considera a taxa de variação do erro ao longo do tempo. Dessa forma, esse componente permite antecipar tendências de alteração do erro e pode contribuir

para reduzir oscilações e melhorar a estabilidade da resposta. Sua contribuição é determinada pelo ganho derivativo kd. Entretanto, como esse termo responde às variações do erro, ele pode ser sensível a ruídos presentes nas medições.

A combinação desses três componentes permite ao PID estabelecer um compromisso entre rapidez de resposta, redução do erro e estabilidade do sistema. A ação de controle do PID na forma contínua pode ser expressa por:

$$u(t) = K_p e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt} \quad (1)$$

Em que e(t) representa o erro do sistema e K_p, K_ie K_d são os ganhos proporcional, integral e derivativo, respectivamente.

O desempenho do controlador depende da escolha adequada dos ganhos kp,ki e kd, uma vez que diferentes combinações desses parâmetros produzem diferentes comportamentos dinâmicos. Assim, a sintonia do PID constitui uma etapa importante para obter uma resposta adequada às características do sistema controlado. Em aplicações envolvendo sistemas móveis, por exemplo, a escolha dos parâmetros deve considerar a dinâmica do robô, a forma como o erro é obtido e as características da ação de controle empregada.

2.5 Plataforma Robótica

O treinamento em aprendizado por reforço pode ser realizado de forma *online* ou *offline*. No treinamento *online*, o agente aprende continuamente a partir da interação direta com o ambiente, atualizando sua política durante a execução. Já no treinamento *offline*, a atualização da política é realizada a partir de um conjunto de transições previamente armazenadas, compostas por estados, ações, recompensas e estados seguintes (s, a, r, s'), sem necessidade de interação contínua com o ambiente durante o processo de aprendizado.

O treinamento *offline* pode proporcionar maior estabilidade e segurança, especialmente em aplicações robóticas, nas quais a exploração direta pode ser custosa ou arriscada (Tang *et al.*, 2025). Por outro lado, essa abordagem pode limitar a capacidade de adaptação do agente a mudanças no ambiente após o treinamento.

O Gazebo é um simulador 3D de robótica de código aberto, amplamente utilizado em pesquisa e desenvolvimento por oferecer um ambiente de simulação baseado em física, permitindo a modelagem de plataformas robóticas e cenários complexos. Sua capacidade de

integrar diferentes sensores, como câmeras e LiDAR, o torna uma ferramenta adequada para o desenvolvimento e a avaliação de estratégias de percepção em robôs móveis. Neste trabalho, o Gazebo é utilizado por permitir a simulação de plataformas robóticas já consolidadas, como o TurtleBot3, em ambientes 3D baseados em física, e sua integração com o ROS 2 e o OpenCV viabilizam testar percepção e controle no ambiente simulado antes da etapa experimental (Koenig e Howard, 2004).

O ROS 2 (*Robot Operating System* 2) é um *framework* de software amplamente utilizado em robótica por fornecer uma arquitetura modular baseada em nós e tópicos, o que facilita o desenvolvimento, a integração e os testes de percepção e controle, além de permitir execução distribuída entre robô e computador. O ROS 2 foi adotado por sua integração com o Gazebo, reduzindo o esforço de implementação e acelerando o ciclo de simulação e validação (Macenski et al., 2022).

O PyTorch é uma biblioteca de código aberto amplamente utilizada no desenvolvimento de modelos de aprendizado de máquina, incluindo algoritmos de aprendizado por reforço profundo. Desenvolvida em Python, fornece ferramentas para construção e treinamento de redes neurais, utilizando grafos computacionais dinâmicos que permitem maior flexibilidade na definição e modificação de arquiteturas. Além disso, disponibiliza funcionalidades para armazenamento, carregamento e reutilização de modelos, facilitando a reprodução de experimentos e o desenvolvimento de soluções baseadas em aprendizado profundo (Paszke *et al.*, 2019).

O Google Colab é um ambiente de notebooks baseado no Jupyter, executado no navegador, que permite o desenvolvimento de aplicações em Python utilizando recursos de computação em nuvem. A plataforma oferece suporte à execução de código, visualização de resultados e integração com bibliotecas amplamente utilizadas em aprendizado de máquina, facilitando a prototipagem, experimentação e compartilhamento de experimentos de forma acessível e reprodutível.

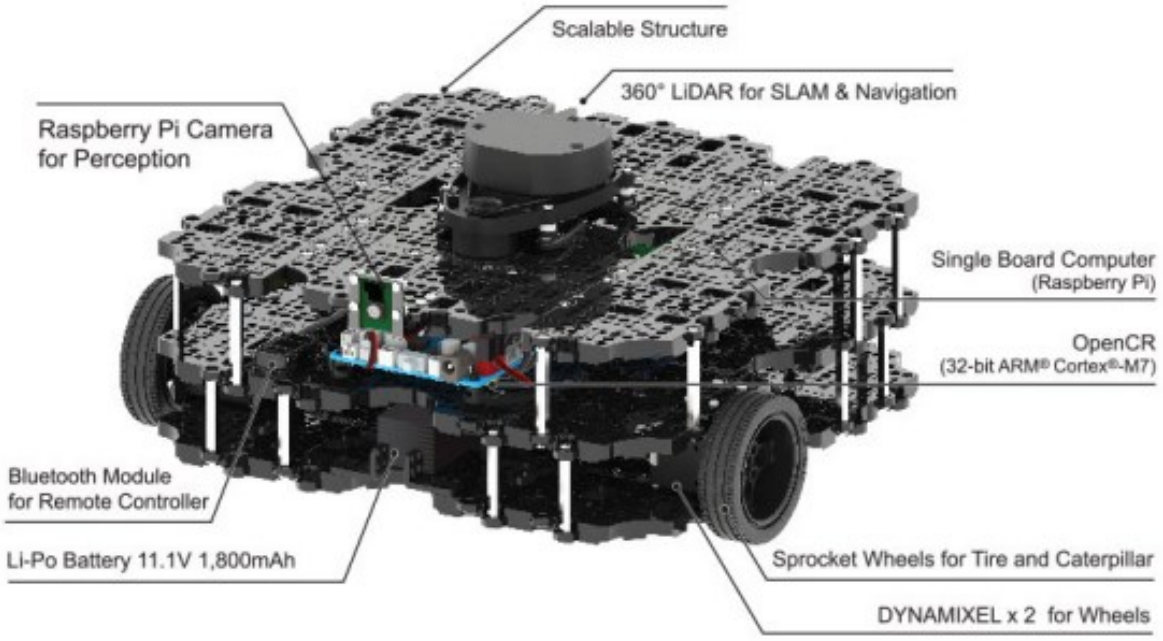
Plataformas robóticas com locomoção diferencial, como as da família TurtleBot, são amplamente utilizadas em pesquisa e ensino em robótica móvel. O TurtleBot3 é um projeto colaborativo que representa a evolução da série TurtleBot, destacando-se pela integração com o ROS 2 e pela disponibilidade de ferramentas e pacotes consolidados (Amsters e Slaets, 2020).

Dentre os modelos disponíveis, destaca-se o TurtleBot3 Waffle Pi, ilustrado na Figura 3. O robô apresenta estrutura compacta, com sensores embarcados e características adequadas para experimentos em ambientes internos. Suas especificações incluem velocidade

translacional máxima de 0,26 m/s, velocidade rotacional máxima de 1,82 rad/s (≈ 104,27°/s), capacidade de carga de até 30 kg e dimensões de 281 × 306 × 141 mm (C × L × A).

Em termos de sensoriamento, o Waffle Pi é equipado com câmera embarcada e unidade de medição inercial (IMU). A câmera permite a obtenção de informações visuais do ambiente, sendo amplamente empregada em tarefas de percepção, enquanto a IMU fornece dados de orientação e aceleração, frequentemente utilizados na estimativa de estado e no controle de robôs móveis.

Figura 3 - Plataforma robótica TurtleBot3 Waffle Pi



Fonte: ROBOTIS (2023).

3 METODOLOGIA

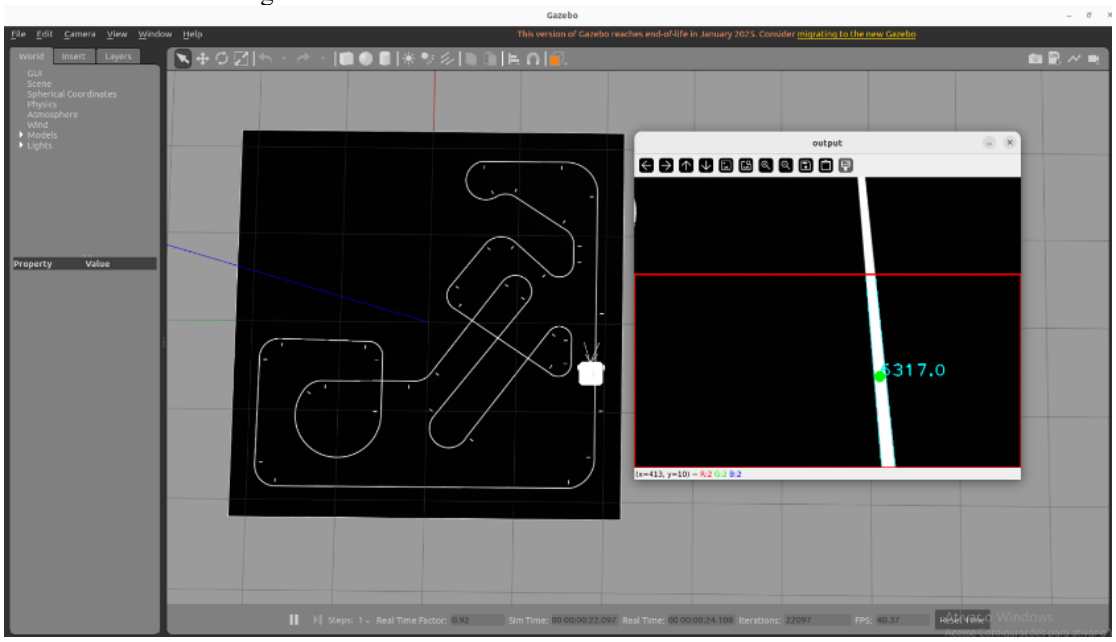
Esta seção detalha o ambiente experimental, a arquitetura do sistema, a modelagem do problema de controle, os controladores avaliados, a configuração de treinamento e os hiperparâmetros utilizados, o ajuste do controlador PID, os critérios de avaliação e seleção, e o protocolo experimental empregado neste trabalho. O objetivo é fornecer uma descrição clara e reprodutível dos métodos e procedimentos adotados para a condução dos experimentos.

3.1 Ambiente Experimental e Arquitetura do Sistema

Os experimentos foram conduzidos em um ambiente computacional configurado com sistema operacional Ubuntu 22.04.5 LTS. A implementação e avaliação dos controladores foram realizadas em um ambiente de simulação utilizando ROS 2 Humble e Gazebo Classic 11.14.0. A linguagem de programação empregada foi Python 3.10.12, com o auxílio das bibliotecas PyTorch (versão 2.7.0) para o desenvolvimento de modelos de aprendizado de máquina, NumPy (versão 1.24.3) para operações numéricas e OpenCV (versão 4.12.0) para processamento de imagens. É importante notar que o ambiente experimental não contou com suporte a CUDA ou cuDNN, o que implica que todos os processamentos foram realizados na CPU.

O Gazebo foi empregado como a principal ferramenta experimental para a implementação e avaliação dos controladores em um ambiente simulado e controlado. Sua utilização permitiu a validação de estratégias de controle, como PID e abordagens baseadas em aprendizado por reforço, de forma reprodutível, garantindo consistência na comparação dos resultados. A integração com o ROS 2 facilitou a comunicação entre os módulos de controle e o ambiente de simulação, permitindo um fluxo de dados eficiente e modular. A Figura 4 apresenta o ambiente no Gazebo juntamente com o primeiro mapa utilizado para os experimentos.

Figura 4 - Plataforma robótica simulada no Gazebo



Fonte: o autor.

O ROS 2 foi utilizado para o controle do robô diferencial, integração dos sensores e implementação das funcionalidades necessárias à simulação. Sua arquitetura baseada em nós possibilitou a organização modular dos componentes e a comunicação assíncrona entre os módulos no ambiente Gazebo. A Tabela 1 descreve as funções desempenhadas por cada módulo.

Tabela 1 - Nós do ROS2

Nó / Pacote	Tópico	Descrição
Gazebo	/gazebo	Ambiente de simulação responsável pela dinâmica física do robô e do cenário virtual
turtlebot3_joint_state	/joint_states	Contém os estados das juntas do robô, incluindo posição e velocidade
robot_state_publisher	/robot_state_publisher	Publica as transformações (TF) do robô com base nos estados das juntas
camera_driver,	/camera/image_raw	Tópico que contém as imagens brutas adquiridas pela câmera
Follower	/action_value	Nó de controle inteligente (ex: DRL) que gera ações a partir das imagens da câmera
manual_controller / follower	/cmd_vel	Tópico de controle de velocidade linear e angular do robô
turtlebot3_diff_drive	/odom	Publica dados de odometria baseados no modelo diferencial do robô
turtlebot3_imu	/imu	Contém dados do sensor IMU, como aceleração e orientação
	/odom	Coleta dados de odometria para análise e armazenamento
data_collector	/action_value	Armazena ações geradas pelo agente
	/error_value	Armazena os erros de controle durante a execução

Fonte: o autor.

O TurtleBot3 Waffle Pi foi adotado como plataforma de referência para os experimentos, com controle baseado em velocidades linear e angular. A velocidade angular foi

limitada a 1 rad/s e a velocidade linear mantida constante para garantir condições padronizadas de avaliação. A câmera embarcada foi utilizada para a percepção do ambiente.

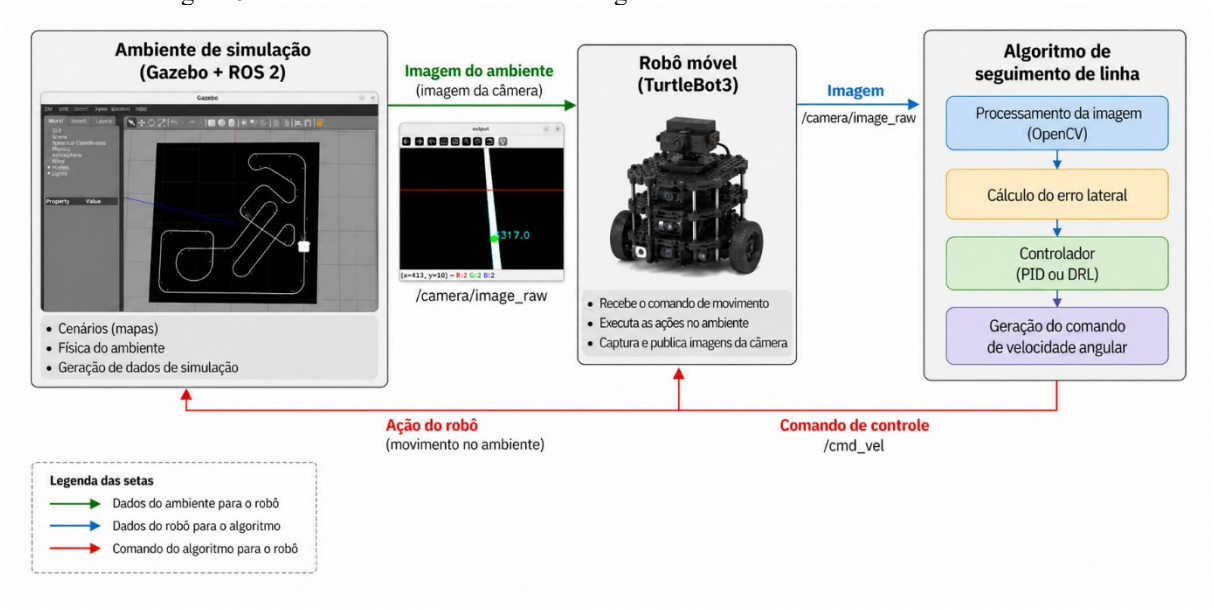
3.2 Percepção e controle

A percepção do ambiente foi realizada exclusivamente por meio da câmera embarcada no robô, sem utilização de sensores adicionais para a identificação da trajetória. O processamento das imagens foi realizado com a biblioteca OpenCV, utilizando uma região de interesse (*Region of Interest*, ROI) predefinida. A identificação da linha de seguimento foi realizada por meio de limiarização por cor, seguida da segmentação binária e da extração dos contornos presentes na imagem. A partir dos momentos geométricos do maior contorno identificado, foi calculado o centróide da linha, cuja coordenada horizontal foi utilizada para determinar o erro lateral. Esse erro corresponde à diferença, em pixels, entre a posição horizontal do centróide da linha detectada e o centro horizontal da imagem. Para utilização pelos controladores, o erro foi limitado ao intervalo de -300 a 300 pixels.

Considerando a resolução horizontal da imagem de 640 pixels e os parâmetros da câmera utilizados na simulação, com altura de 0,25 m, inclinação de aproximadamente 57° em relação à horizontal e campo de visão horizontal de 62,2°, foi possível relacionar o erro no domínio da imagem a uma distância lateral aproximada no plano do solo. A partir do modelo de projeção em perspectiva, o eixo óptico da câmera intercepta o plano do solo a aproximadamente 0,16 m à frente do robô. Nessa posição, um deslocamento de 300 pixels em relação ao centro da imagem corresponde a aproximadamente 0,09 m de deslocamento lateral no plano do solo.

O fluxo de dados do sistema inicia-se com a captura de imagens pela câmera do robô. As imagens são então processadas pelo módulo de percepção visual para identificação da linha e estimação do erro lateral. Este erro é utilizado como entrada para o controlador, que, por sua vez, determina a velocidade angular do robô. A velocidade linear do robô é mantida constante durante todo o processo. A ação de controle (velocidade angular) é então aplicada ao modelo do robô no ambiente de simulação Gazebo, por meio da interface do ROS 2. A Figura 5 apresenta uma representação desse fluxo de dados e das principais etapas envolvidas no ciclo de controle. Este ciclo contínuo permite que o robô ajuste sua trajetória para seguir a linha de forma autônoma.

Figura 5 - Fluxo de dados do sistema de seguimento de linha em ambiente simulado



Fonte: o autor.

O problema de controle de seguimento de linha foi modelado considerando o erro lateral, estimado em pixels no domínio da imagem e limitado ao intervalo de -300 a 300 pixels, como variável de entrada. A variável de saída corresponde à velocidade angular do robô, limitada a 1 rad/s. O objetivo do controlador é minimizar o erro lateral, mantendo o robô alinhado à trajetória de referência durante todo o percurso.

A representação das políticas de controle baseadas em aprendizado por reforço foi implementada por meio de redes do tipo Perceptron Multicamadas (MLP), compostas por duas camadas ocultas de 64 neurônios com ativação ReLU, arquitetura consistente com a prática estabelecida na literatura para espaços de estado e ação de baixa dimensionalidade (Lillicrap *et al.*, 2015). A camada de saída varia conforme a natureza de cada algoritmo: políticas determinísticas (DDPG, TD3, MBRL) produzem diretamente uma ação escalar via função tangente; políticas estocásticas com variância fixa (A2C, PPO) estimam a média da distribuição gaussiana condicionada ao estado; e o SAC, por sua formulação de máxima entropia, estima simultaneamente a média e o desvio-padrão como funções do estado.

A representação do estado foi definida utilizando apenas o erro lateral instantâneo. Essa escolha foi realizada deliberadamente para garantir que todos os controladores, incluindo o PID e os algoritmos de aprendizado por reforço, operassem sob exatamente a mesma informação de entrada, isolando o algoritmo de controle como única variável experimental. Dessa forma, o estudo busca avaliar a capacidade de cada algoritmo em aprender uma política de controle eficaz a partir de uma informação sensorial mínima e comum a todos os métodos, priorizando

a comparabilidade entre as abordagens em vez da maximização do desempenho individual de cada controlador.

3.3 Controladores Avaliados

Esta seção apresenta os controladores avaliados neste trabalho, que incluem tanto um controlador clássico PID quanto diversos algoritmos de aprendizado por reforço profundo. Todos os controladores recebem como entrada o estado do sistema, representado pelo erro lateral, e produzem como saída comandos de velocidade angular para o robô, mantendo a velocidade linear constante.

O Deep Deterministic Policy Gradient foi adotado como um dos controladores avaliados devido à sua adequação a problemas de controle contínuo que exigem ajustes finos e suaves. No contexto do seguimento de linha, essa característica é importante para a correção precisa da trajetória, evitando oscilações e garantindo alinhamento com a pista. A implementação seguiu a arquitetura *actor-critic* com uso de redes-alvo e exploração por ruído nas ações, visando maior estabilidade durante o treinamento (Silver et al., 2014). Os hiperparâmetros e configurações adotados são apresentados na Tabela 2, garantindo a reprodutibilidade dos experimentos não sendo necessárias configurações específicas adicionais para esse algoritmo.

O *Twin Delayed* DDPG foi adotado como um dos controladores avaliados devido à sua maior estabilidade em problemas de controle contínuo, abordando as superestimativas de valor Q que podem ocorrer no DDPG. No contexto do seguimento de linha, essa característica favorece a geração de correções suaves na trajetória, reduzindo oscilações e melhorando o alinhamento com a pista. A implementação considerou a utilização de dois *Critic*, atualização atrasada da política e adição de ruído nas ações-alvo, visando maior robustez durante o treinamento (Lin *et al.*, 2025). Os hiperparâmetros e configurações adotados são apresentados na Tabela 2, garantindo a reprodutibilidade dos experimentos não sendo necessárias configurações específicas adicionais para esse algoritmo.

O *Advantage Actor-Critic* foi adotado como um dos controladores avaliados devido à sua capacidade de aprendizado estável em arquiteturas *actor-critic*. No contexto do seguimento de linha, essa abordagem favorece a adaptação consistente do controle diante de variações na trajetória e perturbações do ambiente. A implementação considerou uma política estocástica e o uso da vantagem (*advantage*) para guiar as atualizações do agente, buscando um equilíbrio entre exploração e explotação (Zhou, Huang e Fränti, 2022). Os hiperparâmetros gerais

utilizados nos experimentos são apresentados na Tabela 2 enquanto os hiperparâmetros específicos do algoritmo são apresentados na Tabela 3, garantindo a reprodutibilidade dos experimentos.

O *Soft Actor-Critic* foi adotado como um dos controladores avaliados devido à sua capacidade de aprendizado robusto por meio da maximização de entropia. No contexto do seguimento de linha, essa característica favorece maior exploração durante o treinamento e contribui para um comportamento mais consistente diante de variações na trajetória e no ambiente. A implementação considerou uma política estocástica com maximização de entropia, visando maior estabilidade e capacidade de generalização. Uma particularidade do SAC é a inclusão de um termo de entropia na função objetivo, o que incentiva o agente a explorar mais o ambiente e a aprender políticas mais robustas e menos determinísticas (Liu, 2024). Os hiperparâmetros e as configurações específicas adotados são apresentados Tabela 5, enquanto os parâmetros comuns a todos os experimentos estão descritos na Tabela 2.

O *Proximal Policy Optimization* foi adotado como um dos controladores avaliados devido à sua robustez e estabilidade em tarefas de controle contínuo. No seguimento de linha, sua estratégia de atualização com limitação por *clipping* favorece ajustes suaves na velocidade angular, evitando variações abruptas e contribuindo para a manutenção da trajetória. A natureza estocástica da política também permite explorar diferentes ações de forma controlada, auxiliando na adaptação a variações do ambiente (Le, Saeedvand e Hsu, 2024). Os hiperparâmetros estão apresentados na Tabela 2 garantindo a reprodutibilidade dos experimentos não sendo necessárias configurações específicas adicionais para esse algoritmo.

O *Model-Based Reinforcement Learning* foi adotado como um dos controladores avaliados devido à sua capacidade de incorporar um modelo do ambiente para apoiar a tomada de decisão. No contexto do seguimento de linha, essa característica permite antecipar o comportamento do sistema e realizar correções mais precisas na trajetória, contribuindo para maior estabilidade durante o controle (Zhu e Zhang, 2021). A implementação considerou o uso de um modelo para predição de estados futuros, auxiliando o processo de aprendizado do agente. É importante esclarecer que a abordagem MBRL aqui utilizada é um método híbrido de aprendizado por reforço baseado em DDPG com incorporação de modelo dinâmico para geração de experiências sintéticas (MBRL-style), e não um MBRL puro. Os hiperparâmetros gerais utilizados nos experimentos são apresentados na Tabela 2, ao passo que as configurações específicas do MBRL são apresentadas na Tabela 4.

3.4 Função de Recompensa

A função de recompensa é responsável por orientar o agente no processo de aprendizado, incentivando a minimização do erro lateral durante o seguimento de linha. Neste trabalho, todos os algoritmos foram treinados utilizando a mesma função de recompensa, garantindo comparabilidade entre os métodos avaliados.

A recompensa foi definida com base no erro lateral, calculado como a diferença entre o centro da linha detectada e o centro da imagem da câmera. A função adotada é dada por:

$$r = 1 - \left(\frac{|\ e\ |}{100}\right)^2 \tag{2}$$

Onde e representa o erro lateral. Essa formulação normaliza o erro pela faixa operacional máxima e aplica uma penalização quadrática, de modo que pequenos desvios resultem em recompensas próximas de 1, enquanto erros maiores são penalizados de forma mais severa. O valor de 100 pixels foi definido empiricamente a partir de testes preliminares realizados no ambiente de treinamento. Diferentes valores foram avaliados para a normalização do erro lateral, sendo adotado o valor de 100 pixels por apresentar comportamento adequado durante o treinamento dos agentes. Esse valor foi utilizado como referência para normalizar a magnitude do erro na função de recompensa.

O valor de 100 pixels foi definido empiricamente a partir de testes preliminares realizados no ambiente de treinamento, correspondendo a aproximadamente um terço do limite máximo de erro lateral definido (300 pixels). Essa escolha faz com que a recompensa se torne negativa para erros acima de 100 pixels, atingindo r = -8 no limite máximo de 300 pixels, o que fornece um sinal de treinamento mais sensível a desvios moderados. Caso o valor de normalização fosse igual ao limite máximo (300 pixels), a recompensa permaneceria próxima de 1 mesmo para desvios consideráveis, reduzindo o incentivo do agente para corrigir o erro lateral antes de se aproximar da condição de terminação do episódio. Diferentes valores foram avaliados para a normalização do erro lateral, sendo adotado o valor de 100 pixels por apresentar o melhor equilíbrio entre sensibilidade da recompensa e estabilidade durante o treinamento dos agentes.

A escolha de uma função quadrática está alinhada com princípios clássicos de controle, favorecendo estabilidade e promovendo uma redução mais eficiente do erro ao longo do tempo

(Kayacan e Chowdhary, 2018). Além disso, a função é contínua e diferenciável, o que contribui para a estabilidade do processo de aprendizado em algoritmos baseados em gradiente.

Não foram incluídas penalizações explícitas sobre as ações de controle, de forma a evitar viés durante o processo de exploração. Ainda assim, comportamentos instáveis são indiretamente desencorajados, uma vez que tendem a aumentar o erro lateral e, consequentemente, reduzir a recompensa acumulada. Dessa forma, a função de recompensa foi projetada para guiar o agente na minimização do erro lateral, promovendo estabilidade e desempenho adequado no seguimento de linha.

3.5 Estratégia Geral de Treinamento

Todos os algoritmos de Aprendizado por Reforço Profundo avaliados neste trabalho foram treinados sob a mesma configuração experimental, garantindo padronização e comparabilidade entre os métodos. O processo de treinamento foi conduzido durante 1000 episódios, sendo que cada episódio possuía duração máxima de 200 passos de interação.

Durante cada episódio, o agente interagia iterativamente com o ambiente de treinamento simplificado, desenvolvido para representar a evolução do erro lateral em função das ações de controle. Nesse ambiente, o estado é representado exclusivamente pelo erro lateral do robô em relação à trajetória, enquanto a ação corresponde à velocidade angular de controle. A evolução do estado é determinada pela ação aplicada ao erro atual, acrescida de uma perturbação aleatória, e o erro lateral é limitado ao intervalo de $[-100,100]$ pixels. O agente observava o estado atual, selecionava uma ação com base em sua política e recebia como retorno um novo estado e uma recompensa associada ao desempenho da ação executada. Esse processo permitia que o agente atualizasse gradualmente sua política de decisão com o objetivo de maximizar a recompensa acumulada ao longo do treinamento.

A duração dos episódios era limitada a 200 passos de interação. Não foi adotado critério de parada antecipada baseado em desempenho *(early stopping)*. Dessa forma, todos os algoritmos foram treinados até o número máximo de episódios previamente estabelecido, garantindo que cada agente tivesse a mesma oportunidade de aprendizado.

3.6 Hiperparâmetros dos Controladores por Reforço

Os hiperparâmetros utilizados nos algoritmos de aprendizado por reforço foram definidos com base em valores consolidados na literatura (Lillicrap et al., 2015; Sutton e Barto,

2018) e refinados empiricamente por meio de experimentação iterativa no ambiente considerado. Parâmetros comuns a todos os algoritmos, como taxa de aprendizado, fator de desconto e tamanho de lote, foram mantidos consistentes entre os métodos, conforme apresentado na Tabela 2.

Tabela 2 - Hiperparâmetros comuns.

Parâmetro	Valor	Descrição
STATE_SIZE	1	Tamanho do vetor de estados
ACTION_SIZE	1	Tamanho do vetor de ações
GAMMA	0,99	Fator de desconto
ACTOR_LR	1×10^{-4}	Taxa de aprendizado da rede Ator
CRITIC_LR	1×10^{-3}	Taxa de aprendizado da rede Crítico
BATCH_SIZE	64	Tamanho do lote de treinamento
MEMORY_SIZE	100000	Capacidade do buffer de replay
TAU	0,005	Taxa de atualização suave
MAX_STEPS	200	Número máximo de passos por episódio
MAX_EPISODES	1000	Número total de episódios de treinamento
ACTION_BOUND	2,0	Limite da magnitude da ação
ACTION_EFFECT	80	Fator de escala da ação no ambiente
NOISE_STD	0,05	Desvio padrão do ruído de exploração

Fonte: o autor.

Os hiperparâmetros específicos de cada algoritmo foram definidos de acordo com as particularidades de cada método. A Tabela 3 apresenta os parâmetros do A2C, incluindo o coeficiente de entropia, o peso da função de valor e a configuração do retorno *n-step*.

Tabela 3 - Hiperparâmetros do algoritmo A2C

Parâmetro	Valor	Descrição
ENTROPY_COEF	0,01	Coef. de entropia
VALUE_COEF	0,5	Peso da função de valor
N_STEPS_RETURN	5	Retorno n-step
NUM_WORKERS	1	Ambientes paralelos

Fonte: o autor.

A Tabela 4 apresenta os parâmetros do MBRL, destacando o horizonte de *rollout*, os passos de treinamento do modelo e o tamanho de lote para o aprendizado do modelo dinâmico.

Tabela 4 - Hiperparâmetros do algoritmo MBRL

Parâmetro	Valor	Descrição
ROLLOUT_HORIZON	5	Horizonte de rollout
MODEL_TRAIN_STEPS	200	Passos de treinamento
MODEL_BATCH_SIZE	128	Tamanho do lote
MODEL_LR	1×10^{-3}	Taxa de aprendizado

Fonte: o autor.

A Tabela 5 apresenta os parâmetros do SAC, com destaque para o coeficiente de entropia e os limites do desvio padrão da política.

Tabela 5 - Hiperparâmetros do algoritmo SAC

Parâmetro	Valor	Descrição
ALPHA	0,2	Temperatura de entropia
LOG_STD_MIN	-20	Limite inferior
LOG_STD_MAX	2	Limite superior

Fonte: o autor.

3.7 Critérios de Seleção do Modelo Ator

Para cada algoritmo, foram realizadas 10 execuções independentes de treinamento com diferentes sementes aleatórias, visando capturar a variabilidade inerente ao processo de aprendizado por reforço e garantir a robustez dos resultados (Arulkumaran et al., 2017)

A seleção do modelo ator utilizado nos experimentos baseou-se em dois indicadores calculados sobre os últimos N=20 episódios de cada execução, correspondentes ao regime estacionário do treinamento. O critério principal foi o RMSE final, definido como:

$$\mathrm{RMSE}_{final} = \sqrt{\frac{1}{N}\sum_{k=T-N+1}^{T} \mathrm{RMSE}_k^2} \tag{3}$$

Onde RMSE_k denota o erro quadrático médio no episódio k, T o total de episódios e N o tamanho da janela de cauda (3) (Fujimoto, Hoof, Van e Meger, 2018). Como critério complementar, foi avaliada a saturação das ações de controle ao final do treinamento, indicando a suavidade da política aprendida. Modelos com menor erro de rastreamento e menor saturação foram priorizados, por demonstrarem equilíbrio entre precisão e suavidade no controle.

3.8 Ajuste do Controlador PID

Para a realização dos experimentos, foi implementada uma versão discreta do controlador PID apresentado no Referencial Teórico. A utilização da forma discreta está relacionada à natureza computacional do sistema de controle, no qual a ação de controle é atualizada em instantes discretos a partir das imagens capturadas pela câmera. No sistema desenvolvido, o ciclo de execução do controlador é acionado por um temporizador com período de amostragem de 60 ms, correspondente a uma frequência de aproximadamente 16,7 Hz.

A partir do erro lateral, são calculados os termos proporcional, integral e derivativo. O termo proporcional utiliza o erro observado no ciclo atual. O termo integral é obtido pelo acúmulo dos erros ao longo dos ciclos de execução, enquanto o termo derivativo é calculado pela diferença entre o erro atual e o erro registrado no ciclo anterior. Assim, a ação de controle utilizada na implementação pode ser representada por:

$$u[k] = -\left(K_p e[k] + K_i \sum_{j=0}^{k} e\,[j] + K_d (e[k] - e[k-1]) \right) \tag{4}$$

em que $e[k]$representa o erro lateral no ciclo k, $e[k-1]$ corresponde ao erro no ciclo anterior, K_p, K_ie K_d representam os ganhos proporcional, integral e derivativo, respectivamente, e $u[k]$ corresponde à velocidade angular de controle. O sinal negativo presente na equação está relacionado à convenção adotada para o cálculo do erro e determina o sentido da correção angular aplicada ao robô.

Os parâmetros do controlador foram definidos a partir de testes preliminares de acompanhamento da trajetória, buscando obter uma resposta capaz de manter o robô próximo à linha de referência, reduzir oscilações e evitar a perda da trajetória. Os valores finais utilizados foram $K_p = 0{,}018$, $K_i = 8 \times 10^{-7}$e $K_d = 3{,}5 \times 10^{-8}$.

A velocidade linear foi mantida constante durante a atuação do controlador, enquanto a velocidade angular foi determinada pelo PID a partir do erro lateral. A ação calculada pelo controlador é posteriormente publicada no tópico de controle do robô, permitindo que a correção da trajetória seja realizada de acordo com a posição da linha detectada na imagem.

3.9 Protocolo Experimental

Sete controladores foram avaliados neste trabalho: o controlador clássico PID e seis algoritmos de aprendizado por reforço profundo: DDPG, TD3, SAC, PPO, A2C e MBRL. Todos os algoritmos DRL foram treinados previamente em ambiente de simulação até apresentarem comportamento estável, resultando em políticas representadas por redes neurais do tipo ator. Após o treinamento, essas políticas foram exportadas e utilizadas exclusivamente em modo de inferência durante os experimentos, sem atualização de parâmetros.

Os experimentos foram conduzidos em três mapas com diferentes níveis de complexidade geométrica. O primeiro cenário consiste em um mapa de maior extensão e complexidade, composto por uma trajetória longa com diferentes tipos de curvatura, exigindo adaptação contínua ao longo do percurso. O segundo corresponde a uma trajetória circular com curvatura suave e constante, representando uma condição controlada para avaliação do comportamento básico dos controladores. O terceiro apresenta uma trajetória em zigue-zague, com curvas curtas e fechadas e mudanças frequentes de direção, impondo maior dificuldade ao controle e exigindo respostas rápidas e precisas.

Após a etapa de treinamento, o modelo de ator selecionado para cada método foi integrado ao sistema de controle no ambiente ROS 2/Gazebo. Durante a execução, o modelo recebia como entrada o erro lateral estimado pelo sistema de percepção visual e gerava como saída o comando de velocidade angular aplicado ao robô. A avaliação foi realizada no mesmo *pipeline* de percepção e controle.

O protocolo considerou dois cenários de velocidade, definidos a partir de experimentos exploratórios com o PID, que determinaram a velocidade máxima estável $v_{\max} = 0{,}25$ m/s. Foram adotados os níveis de 50% ($0{,}125$ m/s) e 75%($0{,}1875$ m/s) de $v_{\max}$. Em cada cenário, cada controlador foi executado em 15 execuções independentes por mapa, registrando erro lateral e ações de controle para análise de precisão, estabilidade e esforço de controle, assegurando comparabilidade entre os métodos.

3.10 Métricas de Avaliação

A avaliação dos controladores foi realizada com base em métricas organizadas em três categorias: precisão, estabilidade e eficiência.

A precisão do seguimento foi avaliada pelo Erro Absoluto Médio (MAE) e pelo Erro Quadrático Médio (RMSE), amplamente utilizados em problemas de rastreamento de trajetória (Leal et al., 2025). O Erro Médio mede a média do erro lateral preservando seu sinal, permitindo identificar tendências sistemáticas de deslocamento do robô em relação ao centro da trajetória:

$$Erro\ Médio = \left(\frac{1}{N}\right)\sum_{i=1}^{N} e_i \tag{5}$$

Onde $e_i$representa o erro lateral no instante i e No número total de amostras. Valores positivos indicam que a linha de referência está deslocada para o lado direito em relação ao centro da imagem, enquanto valores negativos indicam deslocamento para o lado esquerdo.

O MAE mede o desvio médio em relação ao centro da trajetória:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} | e_i | \tag{6}$$

Onde e_i representa o erro lateral no instante i e No número total de amostras. O MAE mede o desvio médio em relação ao centro da trajetória, desconsiderando o sinal do erro.

Enquanto o RMSE penaliza erros maiores de forma mais significativa:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{7}$$

Onde e_i representa o erro lateral no instante ie No número total de amostras. O RMSE é mais sensível a erros de maior magnitude, penalizando desvios acentuados.

A estabilidade foi analisada por meio do desvio lateral máximo (DLM), definido como o valor máximo do erro lateral observado ao longo da execução:

$$DLM = \left(\frac{1}{N}\right)\sum_{J=1}^{N} max\left(e_{\{i,j\}}\right) \tag{8}$$

Onde $e_{i,j}$ representa o erro lateral no instante $i$da execução j, e N representa o número total de execuções. O DLM médio representa o maior desvio lateral médio observado entre as execuções, sendo utilizado como indicador da estabilidade do controlador.

Além do DLM, foi utilizado um índice de robustez, definido a partir da variação do RMSE entre os cenários de velocidade avaliados (Zhang, 2024):

$$RI = 1 - \frac{|\ \Delta\mathrm{RMSE}\ |}{\mathrm{RMSE}_{max}} \tag{9}$$

Onde $\Delta RMSE$representa a diferença entre os valores de RMSE obtidos nos diferentes cenários de velocidade avaliados e $RMSE_{\max}$ corresponde ao maior valor de RMSE observado entre esses cenários. Valores próximos de 1 indicam menor sensibilidade à variação de velocidade e, consequentemente, maior robustez do controlador.

A eficiência foi avaliada pelo esforço de controle, definido como (Ismael, Almaged e Abdulla, 2024):

$$E_{total} = \sum_{i=1}^{N} u_i^2 \tag{10}$$

Onde u_i representa a ação de controle no instante i. Essa métrica penaliza ações de controle excessivas ou abruptas, de modo que menores valores de E_{total} indicam maior eficiência.

Adicionalmente, foram considerados o tempo de execução e a taxa de conclusão da trajetória como métricas de avaliação. O tempo de execução corresponde ao intervalo necessário para que o robô percorra a trajetória definida em cada cenário, sendo contabilizado desde o início do movimento até a conclusão da trajetória ou até a ocorrência de uma condição de falha. Essa métrica permite avaliar a eficiência temporal dos controladores, possibilitando verificar quais métodos são capazes de realizar o percurso em menor tempo. Para cada combinação de controlador, mapa e velocidade, foram realizadas 15 execuções independentes.

A taxa de conclusão da trajetória representa a proporção de execuções em que o robô foi capaz de percorrer toda a trajetória definida no cenário experimental, sem atingir a condição de falha estabelecida. Essa métrica é calculada pela razão entre o número de execuções concluídas e o número total de execuções realizadas, sendo expressa em porcentagem. (Leal et al., 2025).

4 DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento da plataforma experimental utilizada neste trabalho. São descritos o ambiente de simulação, a implementação dos controladores, o fluxo de execução dos experimentos e o processo de aquisição e processamento dos dados, desde o registro das variáveis experimentais até a obtenção das métricas, tabelas e gráficos empregados na comparação entre os controladores. Essas etapas estabelecem a infraestrutura experimental utilizada na avaliação dos métodos de controle investigados.

4.1 Implementação Geral do Sistema

A motivação inicial deste trabalho foi investigar a aplicação de algoritmos de aprendizado por reforço profundo no controle de robôs móveis, em comparação com a abordagem tradicional baseada em controladores PID. Inicialmente, o algoritmo DDPG foi treinado diretamente no ambiente Gazebo. Embora tenha apresentado convergência, o elevado custo computacional da simulação física e do processamento de imagens, aliado à ausência de aceleração por GPU, tornou o treinamento lento e pouco eficiente.

Diante dessa limitação, foi desenvolvido um ambiente sintético de treinamento, capaz de abstrair a dinâmica do erro lateral sem a necessidade de simulação física ou processamento de imagens. Paralelamente, o controlador PID foi sintonizado diretamente no Gazebo, conforme descrito na Seção 3.8, estabelecendo uma referência para a comparação dos métodos.

Após a validação do ambiente sintético com o DDPG, o mesmo ambiente foi utilizado para o treinamento do TD3. Posteriormente, a avaliação foi estendida aos algoritmos SAC, PPO, A2C e MBRL, ampliando o escopo do trabalho para uma comparação sistemática entre diferentes paradigmas de aprendizado por reforço profundo, conforme apresentado na Seção 3.3.

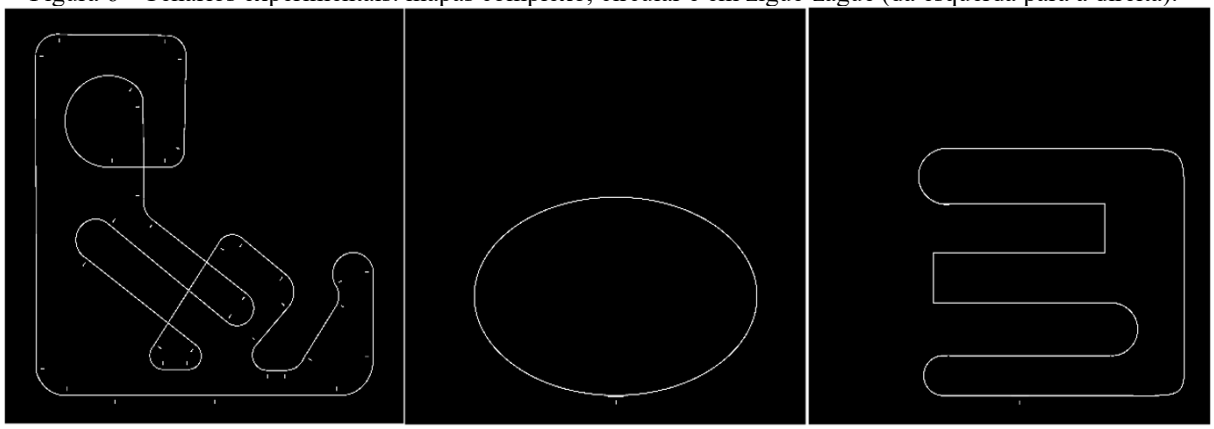
A ampliação do número de algoritmos avaliados exigiu a definição de critérios objetivos para seleção do modelo *Actor* utilizado na etapa de avaliação. Esses critérios, baseados no RMSE final e na saturação das ações de controle, são apresentados na Seção 3.7. A partir dessa estrutura, foram implementados os controladores e conduzidos os experimentos descritos nas seções seguintes.

4.2 Implementação da Plataforma Experimental

O ambiente experimental foi desenvolvido no simulador Gazebo, devido à sua capacidade de simular a dinâmica física do robô e integrar-se ao ROS 2. Essa integração permite a adoção da estratégia *Sim-to-Real*, possibilitando a futura transferência dos controladores desenvolvidos em simulação para uma plataforma robótica real

Para avaliar o desempenho dos controladores, foram desenvolvidos três ambientes de simulação com diferentes níveis de complexidade, ilustrados na Figura 6. Os cenários compreendem um traçado baseado em uma pista de competição de robótica, uma pista circular e um percurso em zigue-zague. Todos foram implementados no Gazebo utilizando apenas a geometria da pista, mantendo constantes a largura da pista e as condições de iluminação, de modo que a geometria do percurso fosse a única variável entre os experimentos.

Figura 6 - Cenários experimentais: mapas complexo, circular e em zigue-zague (da esquerda para a direita).



Fonte: o autor.

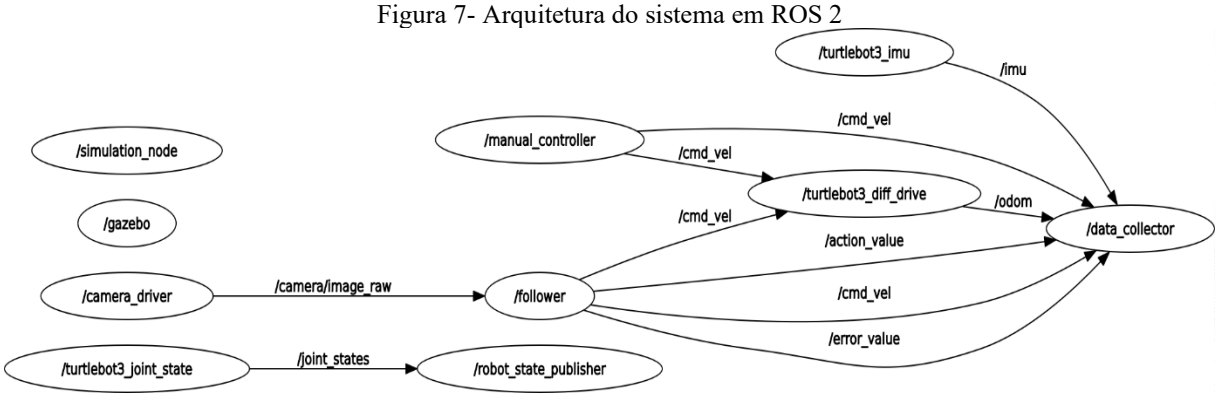
O modelo TurtleBot3 Waffle Pi, apresentado na Figura 3, foi utilizado como plataforma robótica dos experimentos. Com modificações em seu arquivo SDF para adequá-lo à tarefa de seguimento de linha. A câmera embarcada foi reposicionada e configurada com resolução de 640 × 480 pixels, reduzindo o custo computacional do processamento das imagens e mantendo compatibilidade com sistemas reais. O sensor LiDAR foi desabilitado, restringindo a percepção do robô exclusivamente às informações da câmera.

Além disso, a inércia das rodas foi ajustada empiricamente durante os testes preliminares, sendo utilizado um momento de inércia de 0,001 kg·m² nos três eixos. O ajuste foi realizado para reduzir oscilações observadas durante o movimento do robô e proporcionar maior estabilidade à simulação. Essa alteração influencia a resposta dinâmica das rodas às ações

de controle e, por utilizar parâmetros específicos do ambiente simulado, pode gerar diferenças em relação ao comportamento de um robô físico. Essa diferença deve ser considerada como uma limitação para a transferência dos controladores para ambientes reais.

A comunicação entre o ambiente de simulação, o robô e os módulos desenvolvidos foi realizada por meio do ROS 2, que organiza a aplicação em nós independentes responsáveis por funções específicas e interligados por meio de tópicos no modelo de publicação e assinatura (publish/subscribe). Conforme ilustrado na Figura 7, o simulador Gazebo publica as imagens da câmera no tópico "/camera/image_raw", assinado pelo nó "\follower", responsável pelo processamento de visão computacional e pela execução do controlador. Os nós "turtlebot3_diff_drive" e "turtlebot3_imu" disponibilizam, respectivamente, os dados de odometria e da unidade de medição inercial (IMU), enquanto o nó "/data_collector" assina os tópicos de interesse para registrar as informações experimentais.

Figura 7- Arquitetura do sistema em ROS 2



Fonte: o autor.

4.3 Implementação do Ambiente de Treinamento

O treinamento dos algoritmos de aprendizado por reforço profundo foi realizado em um ambiente sintético implementado em Python, em vez do simulador Gazebo, devido ao elevado custo computacional associado ao treinamento no ambiente completo. Como cada algoritmo foi treinado por 1000 episódios, a utilização direta do Gazebo tornaria o processo inviável. O funcionamento do ambiente simplificado, incluindo a representação do estado, a função de recompensa e a dinâmica de transição, foi descrito nas Seções 3.4 e 3.5.

O fator ACTION_EFFECT foi definido empiricamente durante o desenvolvimento do ambiente, adotando-se o valor 80 por proporcionar convergência estável do algoritmo DDPG. Após essa validação, o mesmo ambiente foi utilizado para o treinamento dos algoritmos TD3,

SAC, PPO, A2C e MBRL, garantindo condições experimentais uniformes para todos os métodos avaliados.

Essa estratégia caracteriza um processo de treinamento *sim-to-sim*, no qual a política é aprendida em um ambiente abstrato de baixo custo computacional e posteriormente avaliada no ambiente Gazebo. Embora o ambiente simplificado não represente integralmente a dinâmica do sistema, sua utilização permitiu reduzir significativamente o custo computacional do treinamento, mantendo um protocolo uniforme para comparação entre os algoritmos.

4.4 Implementação dos Controladores

Os controladores baseados em aprendizado por reforço profundo foram implementados em Python utilizando a biblioteca PyTorch, seguindo a arquitetura específica de cada algoritmo. Após o processo de treinamento e seleção do melhor modelo, descritos no Capítulo 3, apenas a rede *Actor* de cada controlador foi utilizada na etapa de execução dos experimentos. Essa escolha deve-se ao fato de que, durante a inferência, a decisão de controle depende exclusivamente da política aprendida, não sendo necessária a utilização das estruturas empregadas durante o treinamento, como a rede Crítico ou o mecanismo de atualização dos parâmetros.

O controlador PID foi implementado em Python e integrado ao nó responsável pelo controle do robô. A cada ciclo de execução, o erro lateral calculado pelo sistema de visão é utilizado como entrada do controlador, que calcula a velocidade angular correspondente a partir das componentes proporcional, integral e derivativa. A velocidade linear foi mantida constante durante todos os experimentos, sendo a velocidade angular a única variável de controle aplicada ao robô. Os ganhos utilizados correspondem aos valores obtidos pelo procedimento de sintonia descrito na Seção 3.8.

O sistema desenvolvido opera em um ciclo contínuo de percepção e atuação executado pelo nó “Follower” durante toda a navegação do robô. Esse ciclo é acionado por um temporizador com período de amostragem de 60 ms. A cada iteração, o nó recebe as imagens publicadas pela câmera embarcada, realiza o processamento de visão computacional para identificar a linha de referência e calcular o erro lateral. Em seguida, esse erro é fornecido ao controlador selecionado, que determina a velocidade angular correspondente. O comando é então publicado no tópico “/cmd_vel”, sendo aplicado ao robô no Gazebo para atualização de sua dinâmica.

Independentemente da estratégia de controle empregada, as etapas de aquisição das imagens, processamento, atuação e registro dos dados permanecem inalteradas, diferenciando-se apenas o controlador responsável por converter o erro lateral na ação de controle. A Figura 8 resume o ciclo de percepção e atuação implementado no nó “Follower”.

Figura 8 - Fluxo de percepção, controle e atuação implementado no nó Follower.

Algorithm 1 Ciclo de percepção e atuação para seguimento de linha

Carregar a configuração do controlador (ganhos do PID ou política treinada π_θ)
Definir a velocidade linear constante $v \leftarrow v_0$
for cada instante de tempo t até o término **do**
  Adquirir a imagem da câmera I_t
  Detectar a linha em I_t e calcular o centróide c_t
  Calcular o erro lateral $e_t \leftarrow c_t - c_{\text{centro}}$
  Limitar o erro $e_t \leftarrow clip(e_t, -300, 300)$
  if controlador = PID **then**
    $\omega_t \leftarrow PID(e_t)$
  else
    $\omega_t \leftarrow \pi_\theta(e_t)$ {Inferência do ator DRL}
  end if
  Normalizar/limitar ω_t aos limites do atuador
  Publicar o comando (v_0, ω_t)
  Registrar (e_t, ω_t, t)
end for=0

Fonte: o autor.

Inicialmente, é carregada a configuração do controlador selecionado e definida a velocidade linear de operação. Em seguida, a cada período de amostragem de 60 ms, a imagem capturada pela câmera é processada para determinar o erro lateral em relação à linha de referência.. Esse erro é utilizado pelo controlador, seja o PID ou a política aprendida pelo algoritmo de aprendizado por reforço, para calcular a velocidade angular aplicada ao robô. Por fim, o comando de velocidade é publicado no “cmd_vel” e as variáveis de interesse são registradas para posterior análise.

4.5 Pipeline de Execução

Durante os experimentos, o nó DataCollector executa paralelamente ao nó Follower, assinando os tópicos publicados pelos demais nós do sistema e registrando continuamente o erro lateral, as velocidades linear e angular, a odometria e os dados da IMU. O registro dessas informações permite reconstruir o comportamento dinâmico do robô ao longo de cada trajetória, possibilitando o cálculo das métricas de desempenho. Ao término de cada percurso, os dados coletados são armazenados em um arquivo no formato CSV para processamento.

Cada um dos sete controladores avaliados (PID, DDPG, TD3, SAC, PPO, A2C e MBRL) foi submetido a testes em três mapas com diferentes níveis de complexidade geométrica (circular, em zigue-zague e complexo) e em duas velocidades lineares de operação (0,125 m/s e 0,1875 m/s). Para cada combinação entre controlador, mapa e velocidade, foram realizadas 15 repetições independentes, totalizando 630 experimentos, cada um com seus respectivos dados armazenados em um arquivo CSV.

Após a aquisição dos dados, um script desenvolvido em Python realiza a leitura dos arquivos CSV e calcula, para cada execução, as métricas de desempenho descritas na metodologia. Em seguida, os resultados são agrupados por controlador, mapa e velocidade, permitindo o cálculo do Índice de Robustez. Por fim, um segundo script gera automaticamente as tabelas utilizadas na comparação entre os controladores, apresentadas no Capítulo 5.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos nos experimentos de seguimento de linha, com foco na comparação entre algoritmos de Aprendizado por Reforço e o controlador PID de referência. A análise considera métricas de desempenho, robustez e comportamento dos controladores em diferentes cenários experimentais. Inicialmente, é apresentada a análise de convergência do treinamento dos algoritmos, utilizada para seleção dos modelos atores empregados nos experimentos.

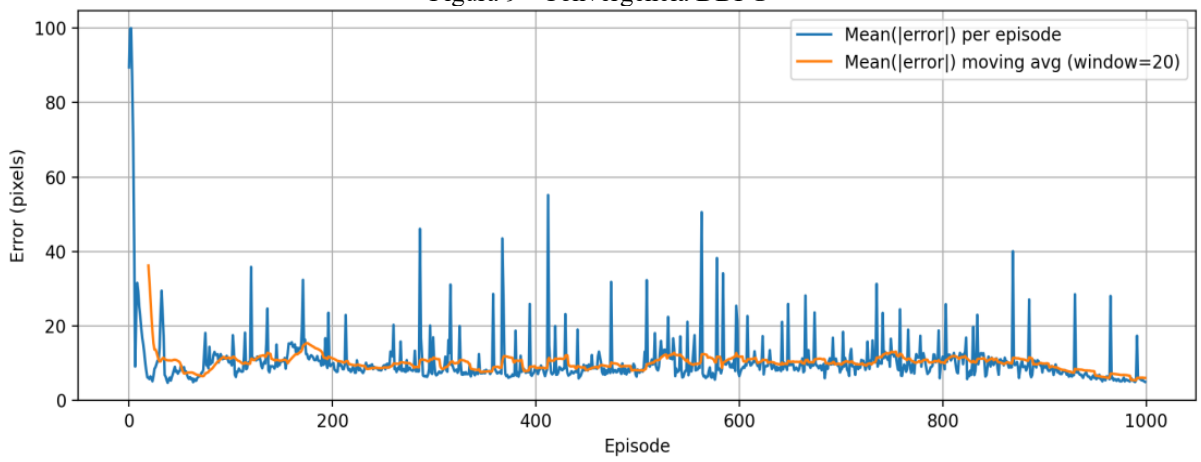
5.1 Análise de Convergência do Treinamento

Esta seção apresenta a análise do comportamento de convergência dos algoritmos de Aprendizado por Reforço durante o treinamento, considerando a evolução do erro médio de rastreamento ao longo dos episódios. Para cada método, são apresentados gráficos do erro por episódio e de sua média móvel, permitindo avaliar aspectos relacionados à estabilidade, redução do erro e comportamento da política aprendida. A partir dessa análise, foi selecionado o modelo ator utilizado nos experimentos de seguimento de linha.

5.1.1 Deep Deterministic Policy Gradient

A Figura 9 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do DDPG, permitindo avaliar seu comportamento de convergência.

Figura 9 - Convergência DDPG



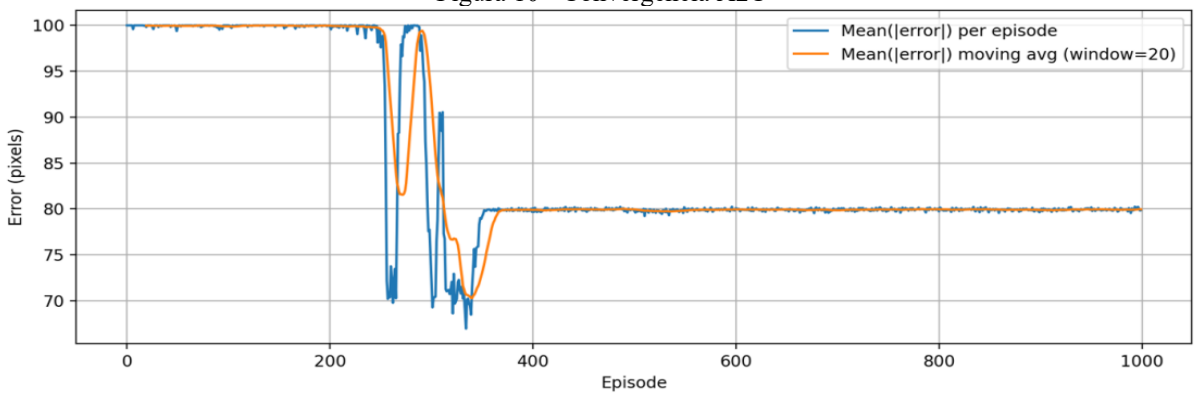
Fonte: o autor.

Observa-se uma redução acentuada do erro nos episódios iniciais, indicando rápida adaptação da política de controle nas primeiras etapas do treinamento. Após essa fase, o algoritmo passa a apresentar comportamento aproximadamente estacionário, caracterizado pela estabilização da média móvel do erro. Apesar da ocorrência de oscilações e picos ocasionais, o comportamento global indica convergência para uma política estável.

5.1.2 Advantage Actor-Critic

A Figura 10 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do A2C, permitindo avaliar seu comportamento de convergência.

Figura 10 - Convergência A2C



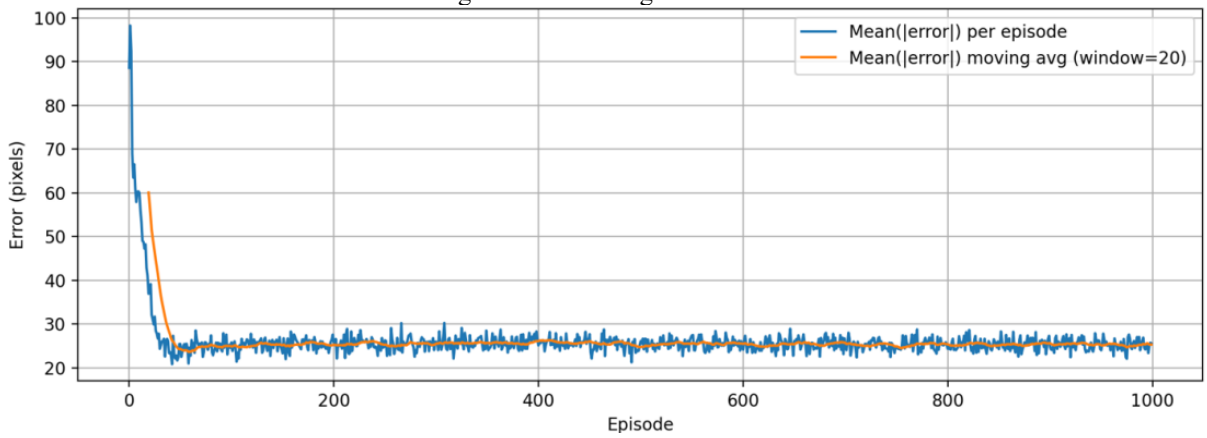
Fonte: o autor.

Observa-se uma fase inicial de transição, seguida por estabilização em regime aproximadamente estacionário. No entanto, o algoritmo mantém níveis elevados de erro ao final do treinamento, além de apresentar frequente saturação dos comandos de controle. Esse comportamento sugere limitações na capacidade de refinamento da política aprendida, resultando em desempenho inferior em relação aos demais métodos avaliados.

5.1.3 Soft Actor-Critic

A Figura 11 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do SAC, permitindo avaliar seu comportamento de convergência.

Figura 11 - Convergência SAC



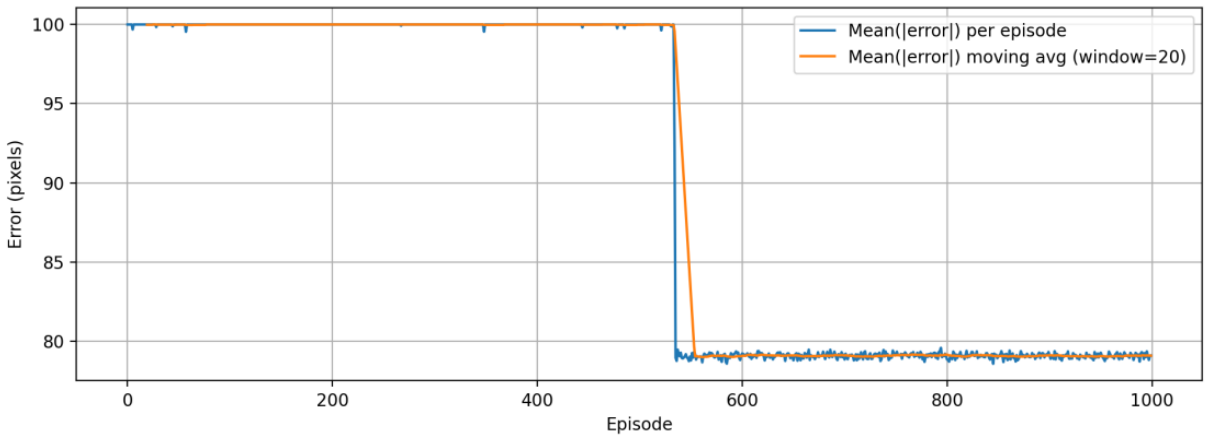
Fonte: o autor.

Observa-se uma redução acentuada do erro nos episódios iniciais, seguida por estabilização em regime aproximadamente estacionário. A baixa incidência de saturação dos comandos de controle ao final do treinamento sugere maior estabilidade da política aprendida, resultando em desempenho mais consistente ao longo dos episódios finais.

5.1.4 Model-Based Reinforcement Learning

A Figura 12 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do MBRL, permitindo avaliar seu comportamento de convergência.

Figura 12 - Convergência MBRL



Fonte: o autor

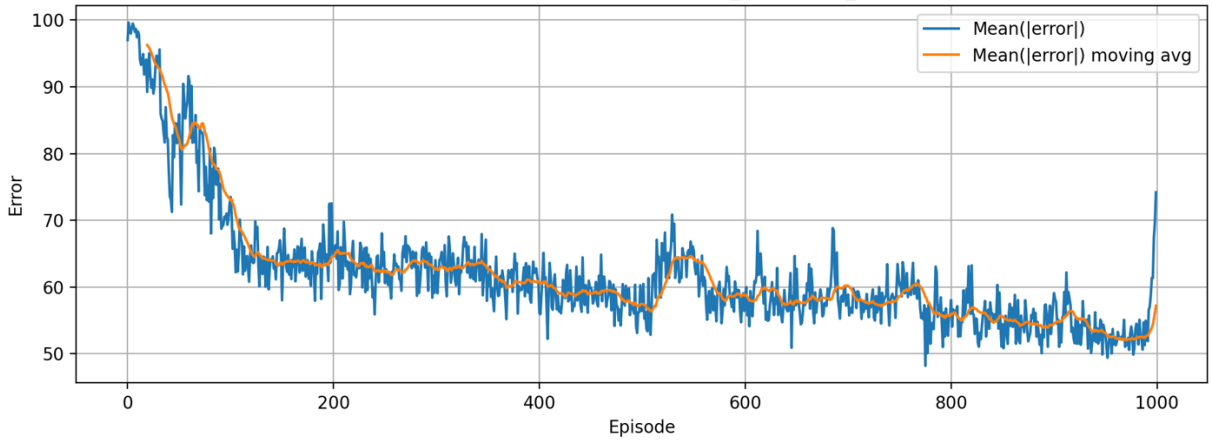
Observa-se um período inicial prolongado com valores elevados de erro, seguido por uma redução mais acentuada e posterior estabilização ao longo dos episódios finais. Esse comportamento sugere que o algoritmo apresenta uma fase inicial mais lenta de aprendizado e atinge um regime aproximadamente estacionário ao final do treinamento. Entretanto, a

estabilização ocorre em níveis de erro superiores aos observados nos demais métodos, indicando limitações no refinamento da política aprendida.

5.1.5 Proximal Policy Optimization

A Figura 13 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do PPO, permitindo avaliar seu comportamento de convergência.

Figura 13 - Convergência PPO



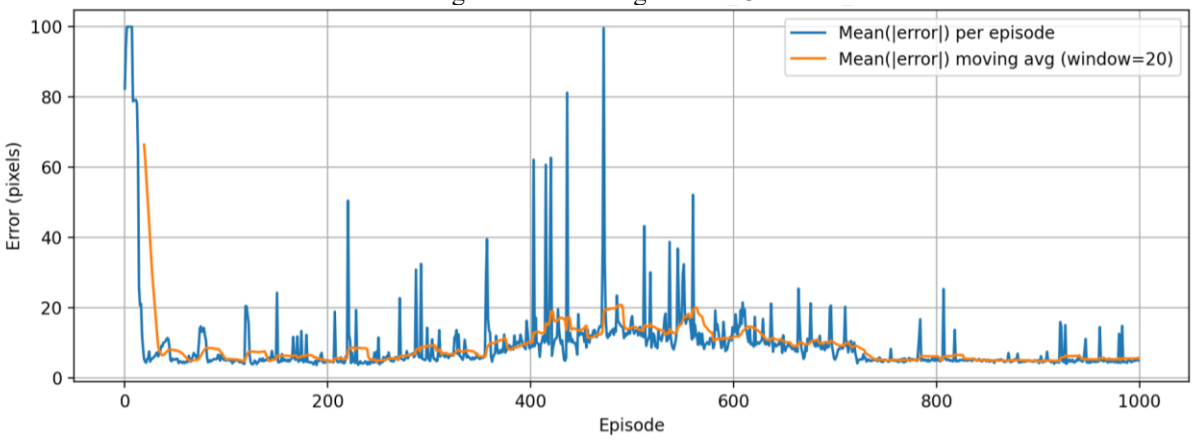
Fonte: o autor.

Observa-se uma redução acentuada do erro nos episódios iniciais, seguida por estabilização em níveis relativamente elevados. Ao longo do treinamento, o algoritmo apresenta oscilações moderadas, sem reduções expressivas adicionais do erro, indicando limitações no refinamento da política aprendida. Além disso, o aumento pontual do erro nos episódios finais sugere instabilidade residual durante o processo de treinamento.

5.1.6 Twin Delayed Deep Deterministic Policy Gradient

A Figura 14 apresenta a evolução do erro médio de rastreamento por episódio ao longo do treinamento do TD3, permitindo avaliar seu comportamento de convergência.

Figura 14 - Convergência TD3



Fonte: o autor.

Observa-se uma redução acentuada do erro nos episódios iniciais, seguida por estabilização em níveis reduzidos. Ao longo do treinamento, são observados períodos de aumento na variabilidade do erro e ocorrência de picos ocasionais, sugerindo instabilidades temporárias durante o processo de aprendizado. Apesar dessas oscilações, o algoritmo apresenta tendência de estabilização ao final do treinamento, indicando convergência para uma política mais consistente.

De forma geral, observam-se diferenças no comportamento de convergência entre os algoritmos avaliados. Métodos como SAC e TD3 apresentaram estabilização mais consistente ao longo do treinamento, enquanto DDPG apresentou maior ocorrência de oscilações intermediárias. O PPO e o A2C apresentaram estabilização em níveis mais elevados de erro, sugerindo limitações no refinamento da política aprendida. Já o MBRL apresentou comportamento distinto, com fase inicial mais lenta e posterior estabilização. Com base nessa análise, foram selecionados os modelos atores utilizados na etapa de avaliação de desempenho.

5.2 Avaliação de Desempenho nos Experimentos

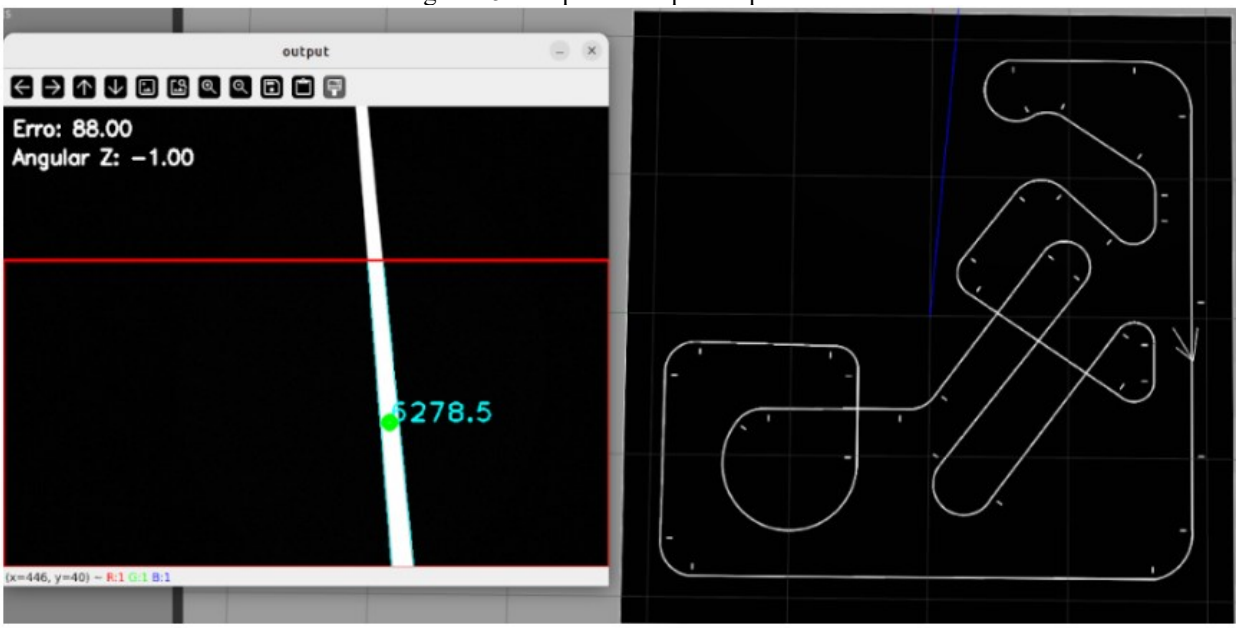
Após a análise da convergência durante o treinamento, foi selecionado, para cada algoritmo de DRL, o modelo ator correspondente ao treinamento que apresentou comportamento de convergência considerado adequado. Esses modelos correspondem às redes neurais resultantes do processo de treinamento e foram utilizados posteriormente na etapa de avaliação de desempenho. Assim, foi utilizado um único modelo ator para cada algoritmo, totalizando seis modelos selecionados: DDPG, TD3, SAC, PPO, A2C e MBRL. A partir desses modelos, foram realizados os experimentos de seguimento de linha em ambiente simulado,

considerando três mapas com diferentes níveis de complexidade e duas velocidades de operação, permitindo a comparação com o controlador PID de referência.

5.2.1 Mapa A – Complexo

O Mapa A, apresentado na Figura 15, representa um cenário de maior complexidade geométrica, caracterizado pela presença de curvas e mudanças de direção que exigem maior capacidade de adaptação dos controladores ao longo do seguimento de linha. Esse cenário permite avaliar o comportamento dos métodos em condições mais exigentes de navegação, destacando diferenças relacionadas à precisão, esforço de controle e capacidade de conclusão da trajetória.

Figura 15 - Mapa A - Mapa complexo



Fonte: o autor.

Conforme apresentado na no Mapa A, observou-se predominância do DDPG em termos de MAE, RMSE e taxa de conclusão de 100%. O PID apresentou valores inferiores de MAE e RMSE em relação aos demais algoritmos de aprendizado, porém com taxa de conclusão de 87%. Em contrapartida, PPO e MBRL apresentaram os maiores valores de MAE e RMSE entre os controladores avaliados.

Tabela 6 - Desempenho para v = 0,125 m/s no Mapa A complexo

Controlador	Erro Médio	MAE	RMSE	DLM	Esforço	Tempo (s)	Taxa (%)
PID	-5,06	16,57	30,41	172,11	41294,50	207,0	87
DDPG	-6,48	17,93	27,85	98,67	41809,30	184,0	100
TD3	-14,24	28,73	43,70	163,67	96893,10	223,0	94
PPO	-5,62	48,11	59,30	186,13	136446,10	223,0	100
SAC	-14,81	30,12	49,71	187,53	86798,87	234,0	100
A2C	-1,69	39,66	49,56	169,07	111333,40	227,0	100
MBRL	-10,82	50,15	62,10	175,93	147018,14	231,0	93

Fonte: o autor.

Os resultados obtidos para o cenário com velocidade de 0,1875 m/s são apresentados na O PID apresentou valores reduzidos de MAE e RMSE, porém com diminuição na taxa de conclusão. Em contrapartida, TD3, PPO, SAC e MBRL apresentaram valores mais elevados de MAE e RMSE, com destaque para o SAC, que obteve a menor taxa de conclusão entre os métodos avaliados. Na Tabela 7 observou-se degradação geral do desempenho dos controladores em relação ao caso anterior, evidenciada pelo aumento do erro de rastreamento e do esforço de controle. O DDPG destacou-se por manter taxa máxima de conclusão, apesar do aumento do RMSE. O PID apresentou valores reduzidos de MAE e RMSE, porém com diminuição na taxa de conclusão. Em contrapartida, TD3, PPO, SAC e MBRL apresentaram valores mais elevados de MAE e RMSE, com destaque para o SAC, que obteve a menor taxa de conclusão entre os métodos avaliados.

Tabela 7 - Desempenho para v = 0,1875 m/s no Mapa A complexo

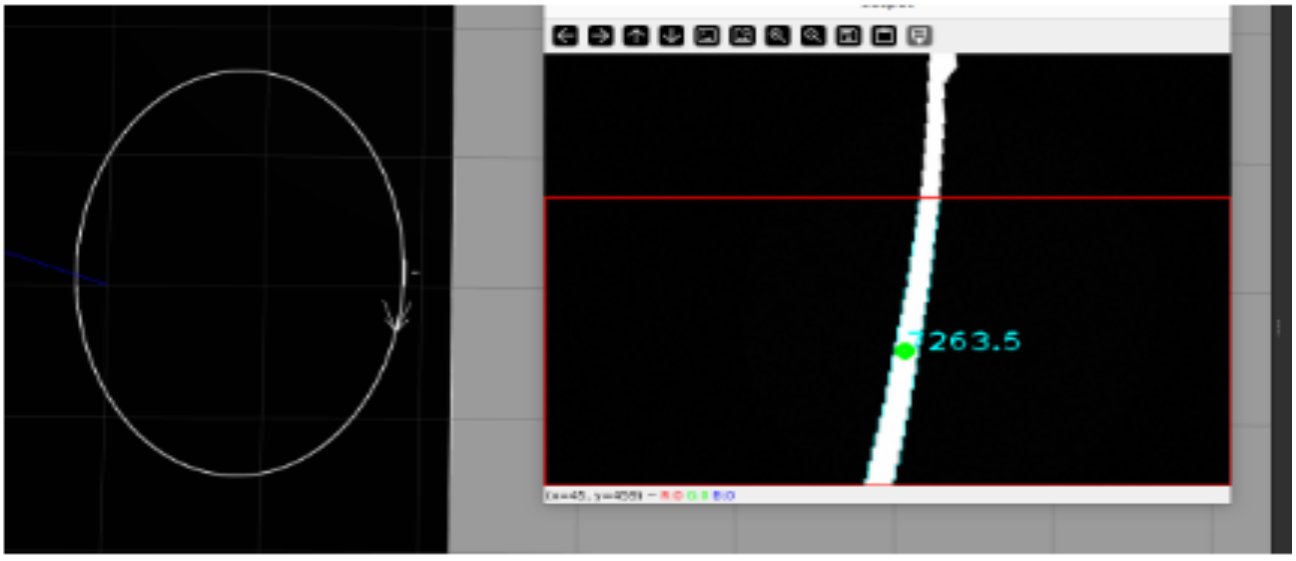
Controlador	Erro Médio	MAE	RMSE	DLM	Esforço	Tempo (s)	Taxa (%)
PID	-6,82	18,59	31,45	151,30	35126,00	150,0	80
DDPG	-10,30	25,93	43,17	159,00	48871,80	148,0	100
TD3	-24,94	48,27	76,67	260,24	95033,30	155,0	100
PPO	-14,59	63,86	80,21	224,30	122775,20	152,0	87
SAC	-24,07	52,45	83,10	272,50	103273,50	162,0	53
A2C	-9,03	53,05	68,72	199,47	103521,80	158,0	100
MBRL	-16,71	64,34	80,46	231,73	124842,30	153,0	100

Fonte: o autor.

5.2.2 Mapa B – Circular

O Mapa B, apresentado na Figura 16 representa um cenário com geometria contínua e menor presença de mudanças bruscas de direção quando comparado aos demais mapas avaliados. Esse cenário permite analisar o comportamento dos controladores em condições de trajetória mais suaves, destacando diferenças relacionadas à precisão de rastreamento e ao esforço de controle.

Figura 16 - Mapa B - Mapa Circular



Fonte: o autor.

Conforme apresentado na Tabela 8, para o cenário de menor velocidade no Mapa B (0,125 m/s), todos os controladores atingiram taxa máxima de conclusão, indicando que o trajeto pôde ser completado independentemente do método empregado. Em termos de precisão de rastreamento, o DDPG apresentou os menores valores de MAE e RMSE, seguido pelo PID com valores semelhantes. TD3 e SAC apresentaram valores superiores dessas métricas em relação aos melhores resultados, enquanto PPO, A2C e MBRL registraram os maiores valores de MAE e RMSE entre os métodos avaliados.

Tabela 8 - Desempenho para v = 0,125 m/s no Mapa B circular

Controlador	**Erro Médio**	**MAE**	**RMSE**	**DLM**	**Esforço**	**Tempo (s)**	**Taxa (%)**
PID	-5,27	5,36	6,59	7,33	4521,60	66,0	100
DDPG	-1,31	5,05	6,49	15,00	4254,67	71,0	100
TD3	-20,43	20,43	22,50	85,53	17290,13	67,0	100
PPO	-6,90	43,47	51,38	125,80	37086,53	67,0	100
SAC	-17,63	17,63	20,80	93,4	14898,53	66,0	100
A2C	-2,57	38,13	45,66	128,00	32472,40	67,0	100
MBRL	-11,47	45,31	54,41	123,00	38772,27	67,0	100

Fonte: o autor.

Os resultados obtidos para o cenário com velocidade de 0,1875 m/s são apresentados na Tabela 9, todos os controladores mantiveram taxa máxima de conclusão, indicando que o aumento da velocidade não comprometeu a capacidade de completar a trajetória. Em termos de precisão, PID e DDPG apresentaram os menores erros de rastreamento, enquanto TD3 e SAC apresentaram aumento mais expressivo do erro. PPO, A2C e MBRL registraram os maiores desvios em relação à trajetória de referência. Em relação ao esforço de controle, os menores valores foram observados para DDPG e PID, enquanto PPO e MBRL demandaram maior intensidade de atuação.

Tabela 9 - Desempenho para v = 0,1875 m/s no Mapa B circular

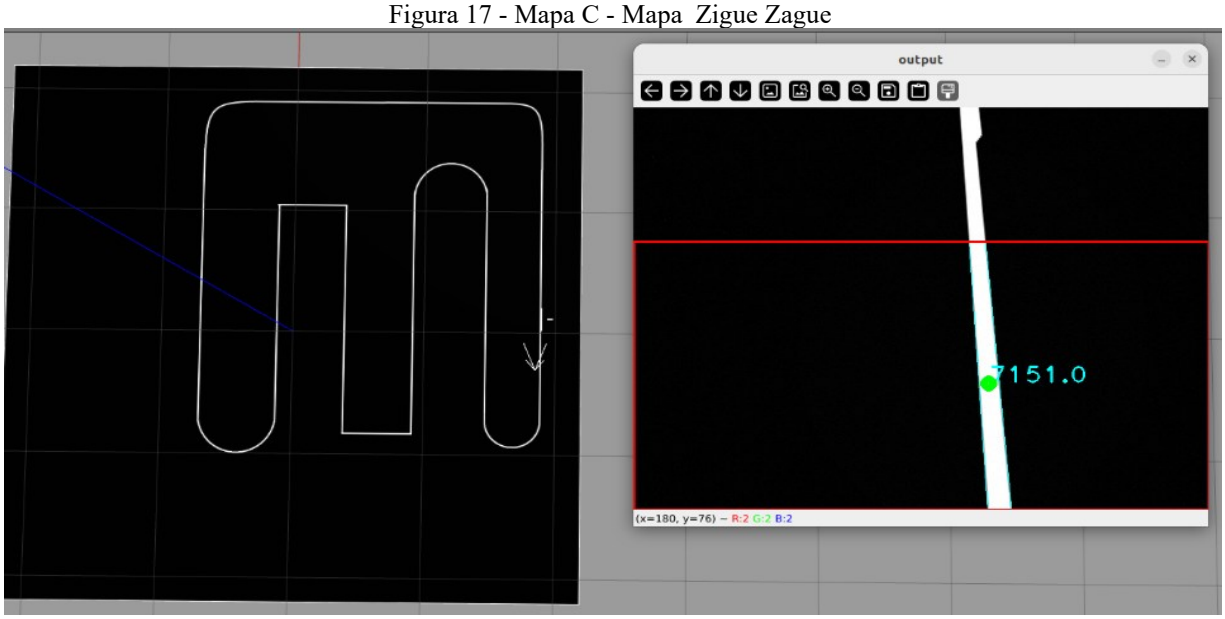
Controlador	Erro Médio	MAE	RMSE	DLM	Esforço	Tempo (s)	Taxa (%)
PID	-8,07	8,18	9,54	8,29	4966,50	48,0	100
DDPG	-4,60	7,13	10,20	20,47	4074,47	45,0	100
TD3	-27,37	27,43	31,34	112,4	15547,87	44,0	100
PPO	-13,92	51,86	62,30	122,40	29767,40	45,0	100
SAC	-25,59	25,62	29,49	109,5	15536,73	48,0	100
A2C	-7,40	42,57	51,40	125,60	24240,33	44,0	100
MBRL	-17,66	54,86	66,53	129,73	31551,67	45,0	100

Fonte: o autor.

5.2.3 Mapa C – Zigue zague

O Mapa C, apresentado na Figura 17, representa um cenário do tipo zigue-zague, caracterizado por mudanças frequentes de direção ao longo da trajetória. Esse cenário permite avaliar a capacidade dos controladores em realizar correções sucessivas durante o seguimento, destacando diferenças relacionadas à precisão de rastreamento, estabilidade e esforço de controle.

Figura 17 - Mapa C - Mapa Zigue Zague



Fonte: o autor.

A Tabela 10 apresenta os resultados obtidos para o Mapa C (0,125 m/s), observou-se elevada taxa de conclusão entre os controladores, com todos os métodos de aprendizado

atingindo 100% e o PID apresentando leve redução nesse indicador. Em termos de precisão de rastreamento, PID e DDPG apresentaram os menores valores de MAE e RMSE, enquanto TD3 e SAC apresentaram valores superiores dessas métricas. PPO, A2C e MBRL registraram os maiores valores de MAE e RMSE em relação à trajetória de referência. PPO, A2C e MBRL registraram os maiores desvios em relação à trajetória de referência. Em relação ao esforço de controle, os menores valores foram observados para PID e DDPG, enquanto PPO e MBRL demandaram maior intensidade de atuação. O tempo de execução apresentou baixa variação entre os métodos avaliados.

Tabela 10 - Desempenho para v = 0,125 m/s no Mapa C Zigue zague

Controlador	**Erro Médio**	**MAE**	**RMSE**	**DLM**	**Esforço**	**Tempo (s)**	**Taxa (%)**
PID	-3,50	9,72	19,04	124,14	17807,00	144,0	93
DDPG	-0,48	12,46	20,53	111,73	22785,73	144,0	100
TD3	-10,99	24,57	40,80	176,13	45948,67	147,0	100
PPO	-1,45	48,02	58,56	184,27	89628,07	147,0	100
SAC	-8,25	23,89	44,17	184,73	47769,60	148,0	100
A2C	2,50	40,78	50,42	163,73	75468,40	147,0	100
MBRL	-5,94	48,58	59,10	182,60	90646,40	146,0	100

Fonte: o autor.

Os resultados obtidos para o cenário de maior velocidade no Mapa C (0,1875 m/s) são apresentados na Tabela 11. Observou-se maior sensibilidade do PID ao aumento da velocidade, refletida na redução da taxa de conclusão, enquanto os demais controladores mantiveram conclusão total da trajetória. Embora o PID tenha apresentado o menor RMSE entre os métodos avaliados, com valor de 27,14, o DDPG apresentou desempenho semelhante em termos de precisão, com RMSE de 31,79, e manteve taxa máxima de conclusão. TD3, PPO, SAC e MBRL registraram maiores desvios em relação à trajetória de referência. Em relação ao esforço de controle, os menores valores permaneceram associados ao PID e ao DDPG, enquanto PPO e MBRL demandaram maior intensidade de atuação. O tempo de execução apresentou baixa variação entre os métodos.

Tabela 11 - Desempenho para v = 0,1875 m/s Mapa C Zigue zague

Controlador	Erro Médio	MAE	RMSE	DLM	Esforço	Tempo (s)	Taxa (%)
PID	-4,41	13,92	27,14	146,23	17139,15	97,0	53
DDPG	-3,99	18,96	31,79	117,63	23336,44	96,0	100
TD3	-15,19	38,27	66,30	249,80	48829,53	100,0	100
PPO	-4,38	61,97	76,68	226,33	79069,93	100,0	100
SAC	-13,23	40,98	73,39	262,93	52617,33	101,0	100
A2C	-2,09	48,39	61,15	191,67	60797,40	106,0	100
MBRL	-9,31	62,16	76,76	226,13	79338,07	100,0	100

Fonte: o autor.

5.3 Comparação de índice de robustez (RI)

A Tabela 12 e a Figura 18 apresentam os resultados do índice de robustez (RI) para os diferentes controladores nos mapas avaliados. Valores mais próximos de 1 indicam menor sensibilidade à variação da velocidade.

Tabela 12 - Comparação do RI dos controladores

Controlador	Mapa A (complexo)			Mapa B (circular)			Mapa C (zigue-zague)		
	$RMSE_{0,125}$	$RMSE_{0,1875}$	RI	$RMSE_{0,125}$	$RMSE_{0,1875}$	RI	$RMSE_{0,125}$	$RMSE_{0,1875}$	RI
PID	30,41	31,45	0,97	6,59	9,54	0,69	19,04	27,14	0,70
DDPG	27,85	43,17	0,65	6,49	10,20	0,64	20,53	31,79	0,65
TD3	43,70	76,67	0,57	22,50	31,34	0,72	40,80	66,30	0,62
PPO	59,30	80,21	0,74	51,38	62,30	0,82	58,56	76,68	0,76
SAC	49,71	83,10	0,60	20,80	29,49	0,71	44,17	73,39	0,60
A2C	49,56	68,72	0,72	45,66	51,40	0,89	50,42	61,15	0,82
MBRL	62,10	80,46	0,77	54,41	66,53	0,82	59,10	76,76	0,77

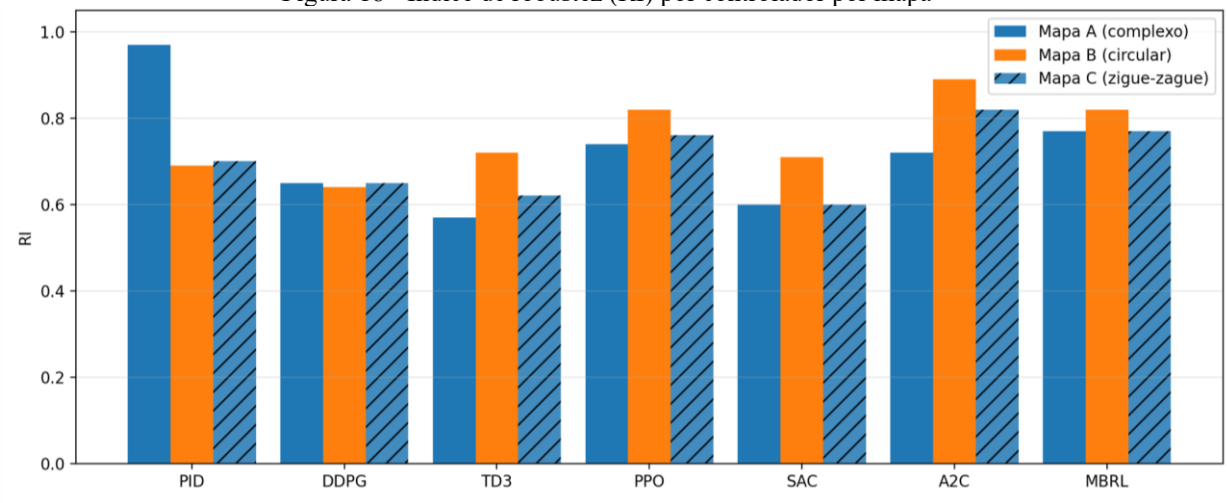
Fonte: o autor.

Observam-se diferenças no comportamento dos controladores entre os cenários experimentais. No Mapa A, o PID apresentou o maior índice de robustez, enquanto os controladores baseados em aprendizado apresentaram maior variação relativa do desempenho com o aumento da velocidade. No Mapa B, destacaram-se A2C, PPO e MBRL, que mantiveram

valores mais elevados de RI. Comportamento semelhante foi observado no Mapa C, no qual A2C e MBRL também apresentaram maiores índices de robustez.

Ressalta-se que valores elevados de RI indicam menor sensibilidade à variação da velocidade, não implicando necessariamente melhor desempenho absoluto.

Figura 18 - Índice de robustez (RI) por controlador por mapa



Fonte: o autor.

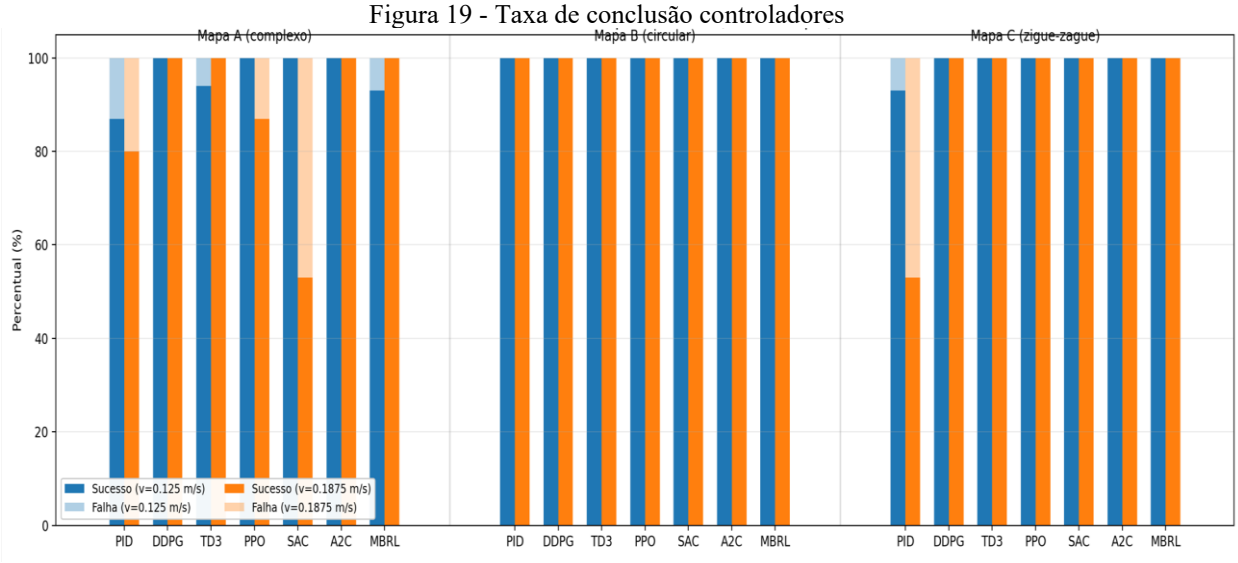
O índice de robustez foi definido com base na variação do RMSE entre as duas velocidades de operação, tendo como objetivo avaliar a sensibilidade do erro de rastreamento ao aumento da velocidade. A taxa de conclusão não foi incorporada diretamente ao cálculo do RI, pois representa uma dimensão distinta do desempenho, relacionada à capacidade de completar a trajetória. Dessa forma, o RI e a taxa de conclusão foram analisados separadamente, permitindo avaliar tanto a sensibilidade do erro à variação da velocidade quanto a capacidade de conclusão da trajetória.

5.4 Taxa de sucesso dos controladores

A Figura 19 apresenta a taxa de sucesso dos controladores nos diferentes cenários avaliados. De forma geral, observa-se elevada capacidade de conclusão para a maioria dos métodos nos mapas B e C em condições de menor velocidade. O aumento da velocidade impactou principalmente o desempenho do PID nos mapas A e C, reduzindo sua taxa de sucesso em relação aos demais controladores. Entre os métodos baseados em aprendizado, DDPG e A2C mantiveram desempenho mais consistente entre os cenários avaliados, apresentando elevadas taxas de conclusão mesmo sob condições mais exigentes. Em contrapartida, alguns

controladores apresentaram maior sensibilidade ao aumento da velocidade, especialmente no Mapa A.

Figura 19 - Taxa de conclusão controladores



Fonte: o autor.

5.5 Análise Comparativa dos Controladores

A Tabela 13 apresenta o ranqueamento dos controladores no Mapa A, adotado como cenário de referência por representar as condições mais exigentes de navegação. Os resultados evidenciam que não existe um controlador absolutamente superior em todas as métricas avaliadas. O PID destacou-se em precisão e esforço de controle em condições mais estáveis, mas apresentou sensibilidade ao aumento de velocidade, com redução da taxa de conclusão. O DDPG mostrou-se a abordagem mais equilibrada, mantendo precisão competitiva e taxa máxima de conclusão em ambas as velocidades. Os demais algoritmos apresentaram erros de rastreamento significativamente maiores, especialmente na velocidade mais elevada, indicando que a escolha do controlador mais adequado depende diretamente dos requisitos da aplicação.

Tabela 13 - Ranqueamento dos controladores por métrica no Mapa A (complexo)

Velocidade m/s	**Métrica**	**1º**	**2º**	**3º**	**4º**	**5º**
v = 0,125	Precisão (MAE)	PID	DDPG	TD3	SAC	A2C
	Precisão (RMSE)	DDPG	PID	TD3	SAC	A2C
	Estabilidade (DLM)	DDPG	TD3	A2C	PID	MBRL
	Esforço Total	PID	DDPG	SAC	TD3	A2C
	Taxa de conclusão	DDPG/PPO/SAC/A2C	TD3	MBRL	PID	
	Tempo de trajeto	DDPG	PID	TD3	PPO	A2C
v = 0,1875	Precisão (MAE)	PID	DDPG	TD3	SAC	A2C
	Precisão (RMSE)	PID	DDPG	A2C	TD3	PPO
	Estabilidade (DLM)	PID	DDPG	A2C	PPO	MBRL
	Esforço Total	PID	DDPG	TD3	A2C	SAC
	Taxa de conclusão	DDPG/TD3/A2C/MBRL	PPO	PID	SAC	
	Tempo de trajeto	DDPG	PID	PPO	MBRL	TD3

Fonte: o autor.

Uma possível explicação para o melhor desempenho geral do DDPG está na configuração utilizada durante seu treinamento, especialmente na função de recompensa e nos hiperparâmetros adotados. A função de recompensa utilizada foi baseada diretamente no erro lateral, incentivando o agente a minimizar o desvio em relação à trajetória de referência. Como a mesma função de recompensa foi utilizada pelos demais algoritmos, não é possível atribuir sua influência exclusivamente ao DDPG. Entretanto, diferenças nos hiperparâmetros e na dinâmica de atualização das redes de cada algoritmo podem ter influenciado o processo de

treinamento e a política obtida. Dessa forma, é possível que a configuração adotada tenha sido mais adequada ao DDPG nas condições experimentais avaliadas, contribuindo para sua maior consistência em termos de erro de rastreamento e taxa de conclusão. Essa hipótese não pode ser confirmada pelos experimentos realizados.

5.6 Limitações

O presente trabalho apresenta algumas limitações que devem ser consideradas na interpretação dos resultados. Os experimentos foram conduzidos exclusivamente em ambiente simulado, sem validação em sistemas reais, e a representação do estado foi limitada ao erro lateral, desconsiderando informações como histórico de erro e orientação do robô. A função de recompensa foi intencionalmente mantida simples para garantir comparabilidade entre os métodos, o que pode ter restringido o aprendizado de comportamentos mais refinados.

Adicionalmente, os hiperparâmetros foram definidos com base na literatura e ajustados empiricamente, sem otimização sistemática. Essa limitação pode ter influenciado o desempenho relativo entre os algoritmos, especialmente TD3 e SAC. Como possibilidade de melhoria, propõe-se a realização de uma otimização sistemática dos hiperparâmetros, especialmente das taxas de aprendizado, do tamanho do lote e dos parâmetros de atualização das redes. Essa otimização poderia ser realizada por meio de busca em grade ou otimização bayesiana, avaliando diferentes configurações durante o treinamento e selecionando aquelas que apresentassem melhor desempenho. Além disso, poderiam ser ajustados parâmetros específicos de cada algoritmo. No TD3, poderiam ser avaliados diferentes valores para o ruído utilizado na suavização das ações-alvo e para a frequência de atualização do ator. No SAC, poderia ser ajustado o coeficiente de entropia, responsável por controlar o equilíbrio entre exploração e aproveitamento das ações. Essas modificações poderiam favorecer maior estabilidade durante o treinamento e melhorar o desempenho dos algoritmos nas condições avaliadas.

Ressalta-se como limitação que o PID foi ajustado diretamente no ambiente de avaliação, enquanto os algoritmos de DRL foram treinados em um ambiente sintético e posteriormente avaliados no Gazebo. Dessa forma, os controladores foram desenvolvidos em contextos distintos de treinamento, aspecto que deve ser considerado na análise comparativa dos resultados.

6 CONCLUSÃO

Este trabalho apresentou uma avaliação experimental comparativa entre seis algoritmos de aprendizado por reforço profundo DDPG, TD3, SAC, PPO, A2C e MBRL e o controlador PID clássico, aplicados ao problema de seguimento de linha por visão computacional em ambiente simulado. Os controladores foram avaliados em três cenários de complexidade geométrica crescente e duas velocidades de operação, considerando métricas de precisão, estabilidade, esforço de controle e robustez.

Os resultados demonstraram que o desempenho dos controladores é fortemente influenciado pela velocidade de operação e pela complexidade geométrica da trajetória. O DDPG destacou-se como a abordagem mais equilibrada, mantendo precisão competitiva e alta taxa de conclusão em todos os cenários avaliados, mesmo com o aumento da velocidade. O PID apresentou resultados consistentes em condições estáveis, com menor esforço de controle, mas demonstrou sensibilidade ao aumento de velocidade, com redução da taxa de conclusão nos cenários mais exigentes. Os demais algoritmos de DRL apresentaram erros de rastreamento significativamente maiores, indicando que a eficácia do aprendizado por reforço profundo depende fortemente da escolha do algoritmo e das condições de operação.

A partir das análises realizadas, verifica-se que os objetivos propostos foram alcançados, uma vez que foi desenvolvido um ambiente de simulação para avaliação dos controladores, implementados e treinados os algoritmos de DRL, definidas métricas quantitativas de desempenho e realizada uma comparação sistemática entre as abordagens. Como principal contribuição, este trabalho fornece uma análise experimental abrangente das condições de aplicabilidade de controladores clássicos e baseados em aprendizado por reforço, evidenciando os cenários em que cada técnica apresenta maior potencial e disponibilizando uma base metodológica que pode apoiar pesquisas futuras em controle inteligente de robôs móveis.

De forma geral, os resultados reforçam o potencial dos métodos de aprendizado por reforço profundo para tarefas de controle contínuo, especialmente em cenários complexos e dinâmicos. Ao mesmo tempo, o desempenho consistente do PID em condições estáveis demonstra que abordagens clássicas permanecem competitivas em sistemas com dinâmica bem conhecida. Como trabalhos futuros, propõe-se a investigação de funções de recompensa mais elaboradas, a otimização sistemática de hiperparâmetros, a avaliação de estratégias híbridas que combinem as capacidades adaptativas do aprendizado por reforço com a estabilidade dos controladores clássicos, e a validação dos métodos em sistemas reais com condições de perturbação externa.

REFERÊNCIAS

AMSTERS, R.; SLAETS, P. Turtlebot 3 as a Robotics Education Platform. Advances in Intelligent Systems and Computing, v. 1023, p. 170–181, 2020. https://doi.org/10.1007/978-3-030-26945-6_16

ARULKUMARAN, K. et al. A Brief Survey of Deep Reinforcement Learning. IEEE Signal Processing Magazine, v. 34, n. 6, p. 26–38, 2017. https://doi.org/10.1007/978-3-030-26945-6_16

BRANCALIÃO, L. et al. Systematic Mapping Literature Review of Mobile Robotics Competitions. Sensors 2022, Vol. 22, Page 2160, v. 22, n. 6, p. 2160, 10 mar. 2022. https://doi.org/10.3390/s22062160

BRIGIDO, W. J. H.; PARENTE DE OLIVEIRA, J. M. The line follower robot: a meta-analytic approach. PeerJ Computer Science, v. 11, p. e2744, 19 mar. 2025. https://doi.org/10.7717/peerj-cs.2744

CARLUCHO, I.; PAULA, M. DE; ACOSTA, G. G. Double Q-PID algorithm for mobile robot control. Expert Systems with Applications, v. 137, p. 292–307, 15 dez. 2019. https://doi.org/10.1016/j.eswa.2019.06.066

FUJIMOTO, S.; HOOF, H. van; MEGER, D. Addressing function approximation error in actor-critic methods. *Proceedings of the 35th International Conference on Machine Learning*, v. 80, p. 1587–1596, 2018.

HAARNOJA, T.; ZHOU, A.; ABBEEL, P.; LEVINE, S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proceedings of the 35th International Conference on Machine Learning*, v. 80, p. 1861–1870, 2018.

ISMAEL, O. Y.; ALMAGED, M.; ABDULLA, A. I. Nonlinear Model Predictive Control-based Collision Avoidance for Mobile Robot. Journal of Robotics and Control (JRC), v. 5, n. 1, p. 142–151, 2024. https://doi.org/10.18196/jrc.v5i1.20615

KAYACAN, E.; CHOWDHARY, G. Tracking Error Learning Control for Precise Mobile Robot Path Tracking in Outdoor Environment. Journal of Intelligent & Robotic Systems 2018 95:3, v. 95, n. 3, p. 975–986, 11 ago. 2018. https://doi.org/10.1007/s10846-018-0916-3

KOENIG, N.; HOWARD, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), v. 3, p. 2149–2154, 2004. https://doi.org/10.1109/IROS.2004.1389727

KRAWCZYK, A. et al. Continual Reinforcement Learning Without Replay Buffers. International IEEE Conference proceedings, IS, n. 2024, 2024. https://doi.org/10.1109/IS61756.2024.10705256

LE, H.; SAEEDVAND, S.; HSU, C. C. A Comprehensive Review of Mobile Robot Navigation Using Deep Reinforcement Learning Algorithms in Crowded Environments. Journal of Intelligent and Robotic Systems: Theory and Applications, v. 110, n. 4, p. 158, 1 dez. 2024. https://doi.org/10.1007/s10846-024-02198-w

LEAL, H. M. et al. Control of a Mobile Line-Following Robot Using Neural Networks. Algorithms 2025, Vol. 18, v. 18, n. 1, 16 jan. 2025. https://doi.org/10.3390/a18010051

LEE, C. T.; SUNG, W. T. Controller Design of Tracking WMR System Based on Deep Reinforcement Learning. Electronics 2022, Vol. 11, Page 928, v. 11, n. 6, p. 928, 16 mar. 2022. https://doi.org/10.3390/electronics11060928

LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 9 set. 2015.

LIN, Y. et al. Efficient TD3 based path planning of mobile robot in dynamic environments using prioritized experience replay and LSTM. Scientific Reports 2025 15:1, v. 15, n. 1, p. 18331-, 26 maio 2025. https://doi.org/10.1038/s41598-025-02244-z

LIU, S. An Evaluation of DDPG, TD3, SAC, and PPO: Deep Reinforcement Learning Algorithms for Controlling Continuous System. p. 15–24, 14 fev. 2024. https://doi.org/10.2991/978-94-6463-370-2_3

LIU, Z. et al. Integrated Task Allocation and Path Coordination for Large-Scale Robot Networks With Uncertainties. IEEE Transactions on Automation Science and Engineering, v. 19, n. 4, p. 2750–2761, 1 out. 2022. https://doi.org/10.1109/TASE.2021.3111888

MACENSKI, S. et al. Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics, v. 7, n. 66, 1 maio 2022. https://doi.org/10.1126/scirobotics.abm6074

MNIH, V. et al. Asynchronous methods for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*, v. 48, p. 1928–1937, 2016. DOI: https://doi.org/10.48550/arXiv.1602.01783

MOERLAND, T. M. et al. Model-based Reinforcement Learning: A Survey. Foundations and Trends in Machine Learning, v. 16, n. 1, p. 1–118, 30 jun. 2020. https://doi.org/10.1561/2200000086

MOHINDRU, P. Review on PID, fuzzy and hybrid fuzzy PID controllers for controlling non-linear dynamic behaviour of chemical plants. Artificial Intelligence Review 2024 57:4, v. 57, n. 4, p. 97-, 23 mar. 2024. https://doi.org/10.1007/s10462-024-10743-0

OGATA, K. Modern control engineering. 5. ed. Boston: Prentice Hall, 2010.

PARK, Y.; JUN, W.; LEE, S. A Comparative Study of Deep Reinforcement Learning Algorithms for Urban Autonomous Driving: Addressing the Geographic and Regulatory Challenges in CARLA. Applied Sciences 2025, Vol. 15, Page 6838, v. 15, n. 12, p. 6838, 17 jun. 2025. https://doi.org/10.3390/app15126838

PASZKE, A. et al. PyTorch: an imperative style, high-performance deep learning library. arXiv, 3 dez. 2019. DOI: https://doi.org/10.48550/arXiv.1912.01703

PAYÁ, L. et al. Deep Reinforcement Learning of Mobile Robot Navigation in Dynamic Environment: A Review. Sensors 2025, Vol. 25, Page 3394, v. 25, n. 11, p. 3394, 28 maio 2025. https://doi.org/10.3390/s25113394

PEREIRA, M. A. et al. A Systematic Literature Review of Autonomous Line-Following Robots using Reinforcement Learning. Revista de Informática Teórica e Aplicada, v. 33, n. 1, p. 50–60, 30 jan. 2026. https://doi.org/10.22456/2175-2745.150105

RAZALI, M. R. et al. A hybrid controller method with genetic algorithm optimization to measure position and angular for mobile robot motion control. Frontiers in Robotics and AI, v. 9, p. 1087371, 12 jan. 2023. https://doi.org/10.3389/frobt.2022.1087371

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. arXiv, 2017. DOI: https://doi.org/10.48550/arXiv.1707.06347

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. Introduction to Autonomous Mobile Robots (Intelligent Robotics and Autonomous Agents series) second edition. Introduction to Autonomous Mobile Robots used in this book, p. Chapter 4, 2004.

SILVER, D. et al. Deterministic policy gradient algorithms. Proceedings of the 31st International Conference on Machine Learning, v. 32, n. 1, p. 387-395, 2014. Disponível em: https://proceedings.mlr.press/v32/silver14.html

SOLANES, J. E.; GRACIA, L. Mobile Robots: Trajectory Analysis, Positioning and Control. Applied Sciences 2025, Vol. 15, Page 355, v. 15, n. 1, p. 355, 2 jan. 2025. https://doi.org/10.3390/app15010355

SOUZA JUNIOR, F. R. DE; COSTA RAMOS, D. Plataforma baseada em ROS2 e Gazebo para avaliação de controladores em robótica móvel com visão computacional e inteligência artificial. E&S Engineering and Science, v. 15, n. 2, 6 jul. 2026. https://doi.org/10.18607/ES20261521680

SOUZA JÚNIOR, F. R. DE; RAMOS, D. C. Control Strategy Comparison for an Intelligent Mobile Mechanism. p. 265–273, 2027. https://doi.org/10.18607/ES20261521680

SUTTON, R. S.; BARTO, A. G. Reinforcement learning: an introduction. 2. ed. Cambridge: MIT Press, 2018.

TANG, C. et al. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. Proceedings of the AAAI Conference on Artificial Intelligence, v. 39, n. 27, p. 28694–28698, 11 abr. 2025. https://doi.org/10.1609/aaai.v39i27.35095

WANG, W. et al. Path Planning Method of Mobile Robot Using Improved Deep Reinforcement Learning. Journal of Electrical and Computer Engineering, v. 2022, p. 5433988, 1 jan. 2022. https://doi.org/10.1155/2022/5433988

YU, X. et al. A Self-adaptive SAC-PID Control Approach based on Reinforcement Learning for Mobile Robots. International Journal of Robust and Nonlinear Control, v. 32, n. 18, p. 9625–9643, 19 mar. 2021. https://doi.org/10.1002/rnc.5662

ZHANG, C. Advancing Robotic Capabilities: The Pivotal Role of PID Control in Modern Robotics. AIP Conference Proceedings, v. 3194, n. 1, 11 dez. 2024. https://doi.org/10.1063/5.0226493

ZHOU, C.; HUANG, B.; FRÄNTI, P. A review of motion planning algorithms for intelligent robots. Journal of Intelligent Manufacturing, v. 33, n. 2, p. 387–424, 1 fev. 2022. https://doi.org/10.1007/s10845-021-01867-z

ZHU, K.; ZHANG, T. Deep reinforcement learning based mobile robot navigation: A review. Tsinghua Science and Technology, v. 26, n. 5, p. 674–691, 1 out. 2021. https://doi.org/10.26599/TST.2021.9010012

ZIEGLER, J. G.; NICHOLS, N. B. Optimum Settings for Automatic Controllers. Journal of Fluids Engineering, v. 64, n. 8, p. 759–765, 1 nov. 1942. https://doi.org/10.1115/1.4019264