UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE SISTEMAS DE INFORMAÇÃO

FABRICIO PINHEIRO DE SOUZA

CONTROLADOR BASEADO EM APRENDIZADO POR REFORÇO PARA ROBÔS SEGUIDORES DE PESSOAS

Itacoatiara – Amazonas
Junho – 2023

FABRICIO PINHEIRO DE SOUZA

CONTROLADOR BASEADO EM APRENDIZADO POR REFORÇO PARA ROBÔS SEGUIDORES DE PESSOAS

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Sistemas de Informação.

ORIENTADOR: PROF. DR. EDSON DE ARAÚJO SILVA

Itacoatiara – Amazonas

Junho – 2023

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Souza, Fabricio Pinheiro de
S729c Controlador baseado em aprendizado por reforço para robôs seguidores de pessoas / Fabricio Pinheiro de Souza . 2023
27 f.: il. color; 31 cm.

Orientador: Edson de Araújo Silva
TCC de Graduação (Sistemas de Informação) - Universidade Federal do Amazonas.

1. Robótica. 2. OpenAI Gym. 3. Deep Q-Network. 4. Controlador. 5. Aprendizado Por Reforço. I. Silva, Edson de Araújo. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Curso de Sistemas de Informação - ICET

FOLHA DE APROVAÇÃO

FABRICIO PINHEIRO DE SOUZA

CONTROLADOR BASEADO EM APRENDIZADO POR REFORÇO PARA ROBÔS SEGUIDORES DE PESSOAS

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em 28 de junho de 2023

BANCA EXAMINADORA

Prof. Dr. Edson de Araújo Silva
Universidade Federal do Amazonas

Prof. Dr. Felipe Gomes de Oliveira
Universidade Federal do Amazonas

Prof. Me. Adriano Honorato de Souza
Instituto Federal do Amazonas

Folha de Aprovação assinada pelo Prof. Dr. Rainer Xavier de Amorim, responsável pela disciplina Trabalho de Conclusão de Curso (Período: 2022.2), onde atesta a defesa do aluno e a presença dos membros da banca examinadora.

Documento assinado eletronicamente por **Rainer Xavier de Amorim**, **Professor do Magistério Superior**, em 30/06/2023, às 18:24, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Edson de Araújo Silva**, **Professor do Magistério Superior**, em 30/06/2023, às 23:01, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Felipe Gomes de Oliveira**, **Professor do Magistério Superior**, em 01/07/2023, às 16:40, conforme horário oficial de Manaus, 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://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1574970** e o código CRC **465BDD1D**.

Rua Nossa Senhora do Rosário - Bairro Tiradentes nº 3836 - Telefone: (92) (92) 99318-2549
CEP 69103-128 Itacoatiara/AM - ccsiicet@ufam.edu.br

Referência: Processo nº 23105.029165/2023-95 SEI nº 1574970

À minha amada avó, que partiu antes de ter a oportunidade de me ver concluir mais essa jornada.

AGRADECIMENTOS

À Deus, o grande programador do universo. Agradeço ao meu orientador pelo seu apoio, orientação e paciência ao longo de todo o processo de pesquisa e redação deste trabalho.

Também quero agradecer à minha família e amigos, pelo seu constante encorajamento e suporte emocional durante essa jornada acadêmica.

Agradeço ainda aos professores e colegas de curso que compartilharam seus conhecimentos e experiências, enriquecendo o meu aprendizado.

Este trabalho não teria sido possível sem o apoio e contribuição de cada um de vocês. Me sinto profundamente grato por todo o suporte recebido ao longo desta jornada acadêmica. Obrigado!

A vitória está reservada àqueles que estão dispostos a pagar o preço.

Sun Tzu

Controlador Baseado em Aprendizado por Reforço para Robôs Seguidores de Pessoas

Fabricio Pinheiro de Souza[1], **Edson de Araújo Silva**[1]

[1]Instituto de Ciências Exatas e Tecnologia – Universidade Federal do Amazonas (ICET/UFAM) - Itacoatiara - Amazonas - Brasil

11235.fps@gmail.com, edsonaraujo@ufam.edu.br

Resumo. *Este trabalho propõe uma técnica de rastreamento e acompanhamento de pessoas por meio de robôs seguidores, utilizando aprendizado por reforço. A abordagem baseia-se no algoritmo Deep Q-Network (DQN), que combina aprendizado por reforço e redes neurais profundas. A implementação consiste em um ambiente de treinamento customizado e um agente capaz de aprender a se locomover a partir de um cenário criado no simulador Gazebo. O trabalho apresenta uma revisão bibliográfica sobre o tema, seguida pela descrição da metodologia utilizada, que envolve pesquisa nas bases de dados, construção da solução e treinamento do modelo. Como resultado, a técnica proposta demonstrou ser viável para ser aplicada e testada em robôs reais e tem potencial para diversas aplicações, como vigilância, busca e resgate, assistência humana e segurança.*

1. Introdução

As pesquisas no campo da robótica para o seguimento de pessoas têm sido o foco de diversos trabalhos devido a seus benefícios em robótica móvel e visão computacional (ALGABRI; CHOI, 2020). O Rastreamento e seguimento de uma pessoa por um robô tem muitas aplicações, como vigilância, busca, resgate, combate e segurança. Pode ser usado também como um assistente humano para transportar ferramentas e equipamentos, além de transmitir imagens de uma pessoa para uma estação central (TAROKH; MERLOTI, 2010).

Diferentes técnicas são propostas para aplicação de seguidores robóticos, dentre elas pode-se citar o trabalho de Schlegel et al., (1998), que usa as informações de contorno e cor da pessoa para rastreamento. Yoshimi et al., (2006) utilizam a detecção de alvos baseados em visão detectando a cor e a textura da roupa da pessoa e sensores ultrassônicos para o robô desviar de obstáculos. Koide e Miura (2016) utilizaram características de cor, altura e marcha. Chen, Sahdev e Tsotsos (2017) apresentam um rastreador baseado em Rede Neural Convolucional (CNN) e visão estéreo. Aye et al., (2019) usam um robô de acompanhamento humano baseado em um algoritmo de rede neural profunda no qual um controlador difuso controla a velocidade do robô e mantém a pessoa alvo na posição central da visão do robô. Han e Peng (2020) propuseram um rastreador baseado em filtro de correlação aprimorado (CF) para rastrear a pessoa alvo. Kulkarni e Pantawane (2022) usam rastreadores baseados em siamês onde é preciso selecionar a pessoa alvo para fornecer as coordenadas para o rastreador que produz o centroide da pessoa rastreada.

A utilização de aprendizado por reforço aplicado ao rastreamento e o seguimento de uma pessoa de forma autônoma por um robô móvel é um campo de pesquisa relati-

vamente novo existindo assim, poucos trabalhos. Embora o uso desta técnica seja comum para a navegação de robôs móveis. A aprendizagem por reforço, juntamente com a aprendizagem supervisionada e não supervisionada formam as três principais abordagens do aprendizado de máquina. Por aprender através da interação contínua com o ambiente que a aprendizagem por reforço destaca-se entre as demais técnicas do aprendizado de máquina, sempre avaliando cada ação em busca de recompensas. Dessa forma, o agente estabelece interação com o ambiente, aprendendo por meio de suas escolhas e recebendo recompensas positivas ou negativas a cada interação, até alcançar seu objetivo (MOREIRA, 2021).

Um algoritmo comumente utilizado no aprendizado por reforço é o *Deep Q-Network* (DQN). O DQN combina o aprendizado por reforço com redes neurais profundas, permitindo que o agente aprenda diretamente a partir de dados de entrada brutos, como imagens. Foi introduzido por Mnih et al., (2013), mostrou-se eficaz em lidar com problemas de alta dimensionalidade e complexidade. Capaz de aprender políticas de controle em diferentes ambientes, mesmo com conhecimento prévio das recompensas e das ações possíveis disponíveis nesse ambiente (WELTER, 2022).

Com o objetivo de atender ao propósito estabelecido, este trabalho adotou um método composto por três etapas. A primeira etapa consistiu em realizar uma pesquisa bibliográfica abrangente nas principais bases de dados disponíveis, a fim de obter embasamento teórico sólido e atualizado. Na segunda etapa, foi realizada a construção da solução que atendesse às necessidades específicas do problema em questão. Por fim, a terceira etapa compreendeu o treinamento do modelo. A adoção desse método permitiu uma abordagem abrangente e sistemática, garantindo a robustez e a eficácia do trabalho realizado.

Diante do exposto e baseado nos trabalhos discutidos, uma técnica de acompanhamento de pessoa baseada em câmera monocular para robôs seguidores de pessoas é proposta. A técnica proposta consiste no treinamento de robôs móveis no simulador Gazebo para ensinar o agente a seguir uma pessoa. O algoritmo usado pelo agente para a tomada de decisões é a do aprendizado por reforço, mais especificamente o algoritmo DQN. Permitindo ao robô tomar decisões eficientes, levando em consideração a interação contínua com o ambiente e a busca por recompensas positivas.

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta alguns conceitos básicos e discute os trabalhos relacionados. A Seção 3 apresenta o método de pesquisa utilizado, a Seção 4 mostra o método proposto, enquanto a Seção 5, os resultados e as discussões. A Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Fundamentação Teórica

Nesta Seção serão apresentados os conceitos relacionados ao tema proposto neste trabalho, além dos trabalhos relacionados ao assunto.

2.1. Conceitos Relacionados

Nesta Subseção são mencionados, de forma sucinta, os principais conceitos utilizados no decorrer deste artigo, considerados essenciais para o entendimento do texto.

2.1.1. Aprendizado por Reforço

A aprendizagem por reforço (RL - do inglês *Reinforcement Learning*) é um aprendizado por meio da interação com o ambiente. Parecido ao modo de adquirir conhecimento humano, ou seja, por meio de tentativas ou erros. Dessa forma, o agente estabelece interações com o ambiente, adquirindo conhecimento por meio da tomada de decisões e recebendo recompensas, sejam elas positivas ou negativas, a cada interação, com o objetivo final de alcançar seu propósito (MOREIRA, 2021).

Todo método que envolva um agente que seja capaz de tomar decisões oriundas da percepção do ambiente em que esteja e que afetem esse estado, com um objetivo relacionado ao ambiente, pode ser considerado um método de aprendizado por reforço. Essa abordagem se diferencia do aprendizado supervisionado, pois não utiliza um conjunto de treinamentos fornecido por um agente externo com conhecimento do problema. Além disso, não pode ser classificada como aprendizado não supervisionado, que busca identificar estruturas escondidas em meio a dados sem rotulação (WELTER, 2022).

O aprendizado por reforço é uma categoria de algoritmos no campo do aprendizado de máquina que consiste em capacitar um agente a adquirir habilidades de comportamento em um ambiente específico, onde o único *feedback* disponível é uma medida numérica de recompensa (WIERING; OTTERLO, 2012). Onde o objetivo do agente é alcançar a maior recompensa numérica ao longo do tempo, sem receber instrução nenhuma sobre quais ações tomar. Ele deve explorar de forma independente todas as ações possíveis e descobrir as recompensas por conta própria (WELTER, 2022).

Além do agente e do ambiente, é possível identificar quatro subelementos principais de um sistema de aprendizado por reforço: uma política, um sinal de recompensa, uma função de valor e um modelo do ambiente. A política determina a maneira como o agente interage com o ambiente e define o perfil de escolha das ações com base nas recompensas recebidas. O sinal de recompensa estabelece o objetivo do problema de aprendizado por reforço, fornecendo uma resposta positiva ou negativa com base na interação do agente com o ambiente, sendo a principal base para a modificação da política de aprendizado. Por outro lado, a função de valor especifica quais escolhas são mais vantajosas a longo prazo, considerando o acúmulo de recompensas ao longo das interações. O último elemento, presente em alguns sistemas de aprendizado, é o modelo do ambiente, que geralmente envolve modificações ambientais para a tomada de decisão (SUTTON; BARTO, 2018).

Os elementos básicos de um sistema de RL apresentados por Vasilev et al., (2019) são:

- **Agente:** a entidade para a qual estamos tentando aprender ações.
- **Ação:** Uma possível resposta ou conjunto de respostas que um agente pode executar. Após cada ação, o ambiente muda de estado e, em seguida, fornece *feedback* ao agente.
- **Recompensa:** o *feedback* que o agente recebe do ambiente após cada ação. O principal objetivo do agente é maximizar o retorno total (recompensas acumuladas) a longo prazo.

- **Estado:** todas as informações disponíveis para o agente sobre seu ambiente atual.
- **Ambiente:** o mundo em que o agente opera.
- **Política:** determina quais ações o agente executará, dado o estado atual. No contexto do aprendizado profundo, podemos treinar uma rede neural para tomar essas decisões.
- **Função Valor:** determina o que é bom para o agente a longo prazo (diferente da recompensa imediata). Ou seja, quando aplicada a função de valor em um determinado estado, ele nos informa o retorno total que podemos esperar no futuro, se começarmos a partir desse estado.

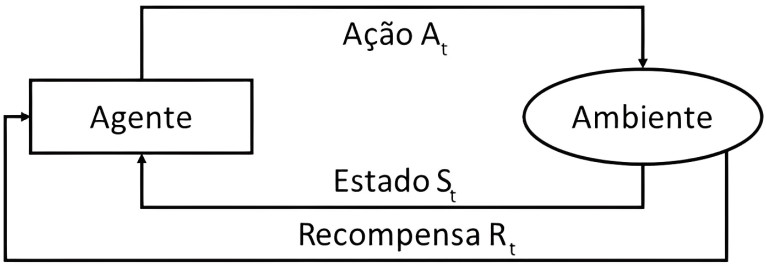

Figura 1. Ciclo de interação entre agente e ambiente

Fonte: Welter (2022)

A Figura 1, representa o ciclo de interação entre o ambiente e o agente. Para definir de forma mais específica as interações entre o agente (robô) e o ambiente, consideramos uma sequência discreta de tempo, t = 0, 1, 2, 3, 4..., na qual o agente recebe uma representação do estado atual do ambiente, S_t, onde $S_t \in S$ que é o conjunto de estados possíveis. Com base nessa representação do estado, o agente seleciona uma ação, $A_t \in A(S_t)$, onde $A(S_t)$ pertence ao conjunto de ações possíveis no estado S_t. Em seguida, como resultado da ação tomada, o agente recebe uma recompensa numérica, $R_{t+1} \in R \subset \mathbb{R}$ e como consequência se encontra em um novo estado, S_{t+1}.

2.1.2. Algoritmo Q-learning

Q-learning é um, algoritmo de aprendizado por reforço que permite a um agente aprender a tomar decisões otimizada em um ambiente desconhecido. O agente aprende a partir de experiências de interação com o ambiente. Ele utiliza uma tabela chamada Q, que armazena os valores Q para cada par estado-ação. Ao realizar todas as ações em todos os estados repetidamente, ele acaba aprendendo as melhores políticas em um contexto geral (WATKINS; DAYAN, 1992).

De acordo com Watkins (1989), *Q-learnig* é uma forma de aprendizado primitivo e livre de modelos, mas funciona como base de dispositivos mais sofisticados. Amplamente utilizado em problemas de controle e tomada de decisão em ambientes complexos, como jogos, robótica e sistemas de recomendação. O algoritmo 1, mostra a forma procedural do Método *Q-learning*.

Algorithm 1 *Método Q-learning*

Inicializa $Q(s, a)$, $\forall s \in S$, $\forall a \in A(\int)$, de forma arbitrária e $Q(\text{terminal}) = 0$
for cada episódio **do**
    Inicializa o estado S
    for cada passo do episódio até o S ser terminal **do**
        Escolhe A de S usando a política de Q
        Toma a ação A, observa R, S'
        Atualiza $Q(S, A) \leftarrow Q(S, A) + \alpha[(R + \gamma \max_a Q(S', a)) - Q(S', A)]$
        Atualiza o estado atual: $s \leftarrow s'$
    end for
end for

Fonte: Adaptado de (SUTTON; BARTO, 2018)

No algoritmo *Q-learning*, uma tabela chamada $Q[s, a]$ é criada na memória para armazenar os valores Q para todas as possíveis combinações de s e a. Essa tabela é usada para determinar a próxima ação a ser executada, selecionando aquela que possui o valor Q máximo para o próximo s' e a'. No entanto, se o número de combinações de estados e ações for muito grande, o uso dessa tabela exigirá muita memória e poder computacional. Isso torna seu uso impraticável em problemas mais complexos do mundo real, especialmente em robótica. Para resolver esse problema, surgiu uma nova abordagem conhecida como *Deep Q-learning* (MNIH et al., 2013). Essa abordagem utiliza redes neurais profundas para aproximar a função de valor Q em vez de armazenar as soluções para cada estado possível.

2.1.3. Algoritmo Deep Q-learning (DQN)

O *Deep Q-learning* ou *Deep Q-network* é uma abordagem moderna para resolver problemas de aprendizado por reforço, que foi introduzido por (MNIH et al., 2013). Neste trabalho eles utilizaram uma rede neural convolucional (CNN) para estimar o valor ótimo "Q"do *Q-learning*, com algumas modificações incorporadas (WELTER, 2022). Q é uma função de valor que estima a recompensa esperada para uma ação tomada em um determinado estado em um ambiente de aprendizado por reforço.

Esses valores Q são então usados pelo algoritmo de aprendizado para determinar a melhor ação a ser tomada em um determinado estado. A junção de aprendizagem profunda e aprendizagem por reforço originou-se o algoritmo *Deep Q-learning* (MNIH et al., 2013), o mesmo usa um aproximador de função de rede neural com pesos θ(*Q-network*) para estimar a função de valor de ação ótima, Q(s, a; θ) $\approx$ Q*(s,a). O algoritmo 2, mostra a forma procedural do Método *Deep Q-learning*.

Algorithm 2 *Deep Q-learning*

Inicializa D para o tamanho N
Inicializa a função Q com pesos aleatórios
for episódio $= 1$ até M **do**
    Inicializa $s_1 = \{x_1\}$ e pré-processa a sequência $\phi_1 = \phi(s_1)$
    for $t = 1$ até T **do**
        Com probabilidade ϵ escolhe uma ação aleatória a_t
        Caso contrário, escolhe $a_t = \max_a Q(\phi(s_t), a; \theta)$
        Executa a ação a_t e observa a recompensa r_t e o estado x_{t+1}
        Define $s_{t+1} = s_t, a_t, x_{t+1}$ e pré-processa $\phi_{t+1} = \phi(s_{t+1})$
        Guarda a transição $(\phi_t, a_t, r_t, \phi_{t+1})$ em D
        Pega uma amostra aleatória de transições $(\phi_j, a_j, r_j, \phi_{j+1})$ de D
        Define $y_j = \begin{cases} r_j & \text{para } \phi_{j+1} \text{ terminais} \\ r_j + \gamma \max_{a'} Q'(\phi_{j+1}, a'; \theta) & \text{para } \phi_{j+1} \text{ não-terminais} \end{cases}$
        Realiza o passo do gradiente descendente em $(y_j - Q(\phi_j, a_j; \theta))^2$
        A cada C passos, restaura $Q' = Q$
    end for
end for

Fonte: Adaptado de (MNIH et al., 2013)

A Figura 2 a seguir ilustra a diferença entre Q-*learning* e *Depp Q-learning* na avaliação do valor Q. O *Deep Q-learning* é uma abordagem que substitui a utilização da tabela Q convencional do *Q-learning* por uma rede neural profunda. Ao invés de armazenar os valores Q em uma tabela que relaciona pares de estado e ação, a rede neural recebe como entrada os estados e mapeia os pares de ação e valor Q correspondente. Essencialmente, a rede neural aprende a representação dos estados e a estimativa dos valores Q para cada ação possível, permitindo uma generalização mais eficiente e flexível do *Q-learning*. O termo Deep Q-network refere-se especificamente à arquitetura da rede neural utilizada no algoritmo DQN (LUU, 2023).

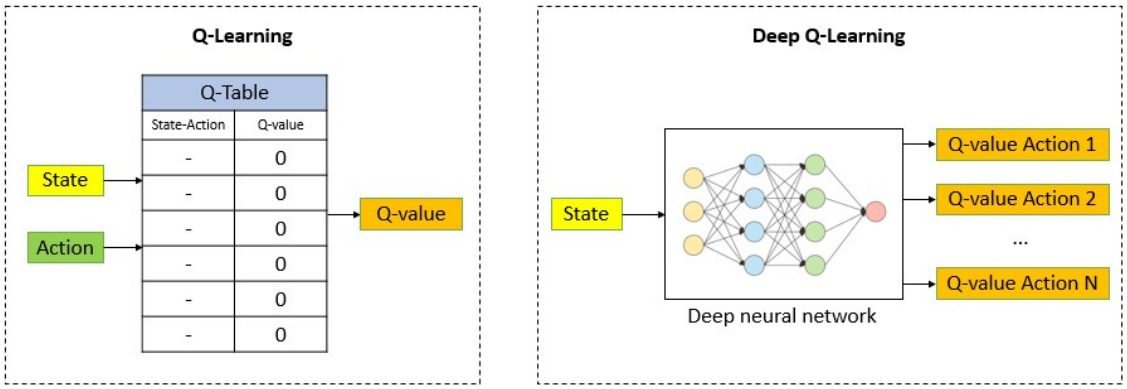

Figura 2. Diferença entre Q-learning e Deep Q-learning

Fonte: Luu (2023)

2.1.4. Gazebo

Gazebo é um simulador de robótica 3D de código aberto que oferece a capacidade de simular com precisão e eficiência robôs únicos e múltiplos, para ambientes internos e externos complexos. Ele oferece um motor de física robusto, gráficos de alta qualidade e interfaces programáticas e gráficos convenientes ao seu dispor tudo isso para simular antes de criar (Open Source Robotics Foundation, 2023).

Amplamente utilizado pela comunidade global de robótica para uma ampla gama de finalidades. Ele segue uma abordagem altamente modular para fornecer os quatro principais componentes necessários para a simulação de robôs (BINGHAM et al., 2019):

1 - Para lidar com forças de colisão, contato e reação entre objetos rígidos, o Gazebo oferece suporte ao uso de vários motores de física.

2 - O Gazebo possui uma vasta biblioteca de sensores comumente encontrados em robôs, como câmeras, lasers, sonares, Sistema de Posicionamento Global (GPS) e Unidade de Medição Inercial (IMU), além de modelos de ruído padrão que podem ser personalizados conforme necessário.

3 - Ele também suporta várias interfaces que permitem aos usuários interagir com a simulação por meio de programação, incluindo a capacidade de escrever *plugins* em *C++*, um sistema de transporte de rede personalizado e o uso de mensagens do *ROS* (Sistema Operacional de Robô).

4 - Além de incluir uma interface gráfica que permite explorar e manipular um ambiente simulado em 3D.

Os simuladores desempenham um papel essencial na área de pesquisa em robótica, permitindo testes rápidos e eficientes de novos conceitos, estratégias e algoritmos. Essas ferramentas de simulação de robótica são utilizadas para desenvolver e testar aplicações embarcadas para robôs sem a necessidade de ter o robô físico, resultando em economia de custos e tempo (QIAN et al., 2014).

2.1.5. Sistema Operacional de Robô (ROS)

O sistema operacional de robô (ROS - do inglês *Robot Operating System*) é um kit de desenvolvimento de código aberto desenvolvido pela *Open Robotics* para aplicações robóticas. O ROS oferece uma plataforma de software padrão para desenvolvedores em todas as indústrias que os levarão da pesquisa e prototipagem até implementação e produção. O projeto produziu um vasto ecossistemas de software para robótica, alimentando uma comunidade global de milhões de desenvolvedores e usuários que contribuem e melhoram esse software (ROS, 2023b).

Embora tenha “Sistema Operacional de Robô” no nome, o ROS não é realmente um sistema operacional. Em vez disso, é um SDK (kit de desenvolvimento de software) que oferece os componentes essenciais para construir suas aplicações robóticas. Seja para um projeto de aula, um experimento científico, um protótipo de pesquisa ou um produto final, o ROS irá ajudá-lo a alcançar seu objetivo de maneira eficiente (ROS, 2023a).

O ROS é um framework distribuído amplamente usado na robótica devido as suas

vantagens, como a facilidade de abstração de hardware e a capacidade de reutilização de código. Ele se baseia em nós, mensagens, tópicos e serviços. O ROS possui um Master que é o núcleo do sistema, sem ele os nós não seriam capazes de de comunicarem. Os nós são arquivos executáveis que se comunicam entre si passando mensagens. Uma mensagem é simplesmente uma estrutura de dados. Um nó publica uma mensagem em um tópico. Esse tópico é usado para identificar o conteúdo da mensagem. Pode ter vários editores e assinantes para um único tópico e um nó pode publicar ou assinar vários tópicos. A solicitação ou resposta é realizada por meio de serviços, que são definidos por uma par de mensagens: uma para solicitação e outra para resposta (QIAN et al., 2014).

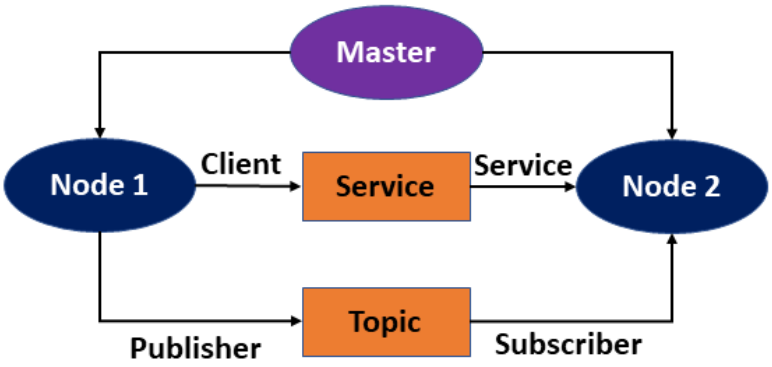

Figura 3. Comunicação no ROS

Fonte: Adaptado de Joseph (2018)

A Figura 3, ilustra o funcionamento da comunicação no ROS. O ROS master desempenha o papel de armazenar e gerenciar todas as informações dos nós. Sua função é permitir que cada nó individual localize os demais nós. Após isso, a comunicação ocorre diretamente entre os nós.

2.1.6. OpenAI Gym

O *OpenAI Gym*, é um kit de ferramenta para desenvolvimento e comparação de algoritmos de aprendizagem por reforço (RL). Ele é compatível com algoritmos escritos em qualquer *framework* e seus ambientes são desenvolvidos em Python (OPENAI, 2023).

O objetivo deste conjunto de ferramentas é permitir a integração da interface de programação de aplicativos (API) do Gym com hardware robótico, possibilitando a validação de algoritmos de aprendizado por reforço em ambientes reais. Isso é alcançado ao combinar o simulador Gazebo com o ROS. Dessa forma, é possível realizar operações no mundo real para testar e aprimorar os algoritmos de aprendizado por reforço (ZAMORA et al., 2016).

O *OpenAI Gym* concentra-se no contexto de aprendizado por reforço episódico, onde a experiência do agente é dividida em uma série de episódios. Em cada episódio, o agente começa em um estado inicial que é selecionado aleatoriamente de uma distribuição, e a interação continua até que o ambiente atinja um estado terminal. O objetivo no aprendizado por reforço episódico é maximizar a expectativa da recompensa

total por episódio e obter um alto nível de desempenho com o menor número possível de episódios (BROCKMAN et al., 2016).

Portanto, a combinação do *OpenAI Gym*, Gazebo e ROS permite aos desenvolvedores treinar e testar agentes de aprendizado por reforço em ambientes virtuais realistas, aproveitando a simulação precisa do Gazebo e as capacidades de controle do ROS. Isso facilita a validação e o aprimoramento de algoritmos de aprendizado por reforço em cenários robóticos antes de implantá-los em um robô real. No *OpenAI gym*, os ambientes são projetados para interagir com o ROS, que atua como uma ponte entre o Gym e o simulador Gazebo (ZAMORA et al., 2016).

2.2. YOLOv4

O método YOLO (do inglês - *You Only Look Once*) foi proposto por (REDMON et al., 2016) é uma ferramenta de Visão Computacional que tem se destacado e recebido considerável atenção nos últimos anos. Desde seu lançamento em 2015, o YOLO foi prontamente reconhecido como uma técnica inovadora de detecção de objetos. Sua abordagem revolucionária permitiu alcançar níveis de precisão iguais ou até superiores aos métodos de detecção existentes na época, porém com uma velocidade de detecção significativamente mais rápida (ALVES, 2022).

Segundo Araújo et al.(2022), o diferencial do YOLO como o próprio nome sugere, a rede olha uma única vez para a imagem, fazem dele um dos algoritmos mais rápidos de detecção já implementados. Capaz de detectar objetos em tempo real de até 30 fps (do inglês - Quadros por Segundo) (REDMON et al., 2016).

A arquitetura do YOLO é composta por 3 estágios: *Backbone*, *Neck* e *Head*, como pode vista na Figura 4. O *backbone* é responsável por extrair características e padrões das imagens de entrada. Ele geralmente consiste em várias camadas convolucionais empilhadas. O *backbone* é projetado para capturar informações de diferentes escalas e níveis de abstração, permitindo que o modelo detecte objetos em várias resoluções e contextos. O papel do *Neck* é coletar mapas de características de diferentes estágios. E por último tem-se o *Head* ou *Dense Prediction*, responsável por gerar as predições de detecção, contendo um vetor com as coordenadas da caixa delimitadora detectada (centro, altura, largura), a pontuação de confiança e o rótulo (ARAÚJO et al., 2022).

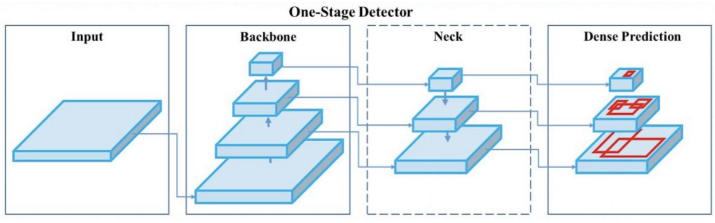

Figura 4. Arquitetura do YOLO

Fonte: Adaptado de Bochkovskiy, Wang e Liao (2020)

Foi escolhido a versão do YOLOv4 para este trabalho por ser a mais rápida na detecção de objetos. Ela foi projetado com foco no desempenho e na eficiência computacional, bsucando melhorar a velocidade em comparação com versões anteriores como o YOLOv3.

2.3. Trabalhos Relacionados

Hu, Zhang e Li (2022) têm o objetivo de propor uma nova estratégia de acompanhamento de guia preditivo para melhorar a estabilidade de distância astronauta-robô em ambientes obstrutivos. Combinando um navegador de aprendizado por reforço profundo e um preditor baseado em filtro de *Kalman* para gerar sequências de movimento otimizadas para seguir com segurança o astronauta e obter orientação preditiva sobre os futuros movimentos do astronauta. Foram realizadas simulações para avaliar a adaptabilidade do robô a ambientes complexos desconhecidos, testes comparativos sugerindo que esta estratégia superou as outras duas em termos de estabilidade e seguimento e experimentos com robô físico simulando um ambiente real em Marte. O modelo proposto obteve uma taxa de sucesso de 95% em tarefas de navegação simuladas e bem adaptadas a ambientes complexos não treinados e configurações variadas de movimento de robôs.

Embora ambos os trabalhos explorem o uso de aprendizado por reforço e envolvam o seguimento de alvos, suas abordagens, objetivos e resultados específicos são distintos. Este trabalho se diferencia de (HU; ZHANG; LI, 2022) por usar câmera monocular para detectar a pessoa e com base nisso realizar o treinamento do agente no simulador para aprender a toma decisões eficientes e adaptáveis de acordo com o ambiente em que está interagindo.

Em Kulkarni e Pantawane (2022) os autores apresentam o rastreador baseado em siamês. As imagens são fornecida diretamente ao rastreador onde o mesmo produz o centroide da pessoa alvo. Por meio do centroide e profundidade estimada da posição, descobre-se a distância e o ângulo entre o robô e a pessoa que por meio do navegador produz a velocidade linear e angular para controlar o robô. Para detectar a pessoa utilizou-se o modelo de aprendizagem profunda: YOLOv5 que por meio de uma câmera estéreo fornece imagens vermelha, verde, azul e imagens de profundidade (RGBD). O rastreador se mostrou bem sucedido e preciso. O método proposto funciona muito bem em situações desafiadoras com uma taxa de quadros de 30 fps e em oclusões parciais.

A diferença deste trabalho para o de (KULKARNI; PANTAWANE, 2022), é que este utiliza aprendizado por reforço para treinar o agente em ambientes simulado para aperfeiçoar suas habilidades. Embora ambos os trabalhos abordam o rastreamento de pessoas, eles diferem na abordagem de detecção (YOLOv4 vs. YOLOv5) e na utilização de informações de profundidade. Em Kulkarni e Pantawane(2022) utilizam-se do aprendizado profundo pelo fato de fazer parte da arquitetura do YOLOv5.

Zhang et al. (2019), propuseram um guia seguidor de alvos para um robô móvel baseado em visão composto em três partes: a parte de rastreamento visual; a parte de redetecção do alvo; e a parte servo visual. Uma estratégia de amostragem baseado na banda de contorno do alvo (TCB - do inglês *Target Contour Band*) foi proposta para melhorar o desempenho do rastreador. Experimentos abrangentes na plataforma do robô foram conduzidos em ambientes internos e externos para testar o guia. No experimento externo o robô seguiu o alvo de forma robusta por cerca de 648 metros, validando o guia seguidor de alvo. O sistema é projetado para ser aplicado em ambientes dinâmicos, onde o alvo pode se mover e o robô precisa se adaptar às mudanças de posição.

Este trabalho se diferencia ao de (ZHANG et al., 2019), por usar aprendizado por reforço no treinamento do agente no ambiente simulado e para detecção da pessoa alvo

utilizou-se do YOLOv4. No trabalho de Zhang et al., (2019), utiliza-se de um método de triagem de recursos baseado em TCB para distinguir a área alvo e a área de fundo.

3. Método da Pesquisa

Com o intuito de alcançar o objetivo estabelecido neste trabalho, foi realizado o seguinte método apresentado na Figura 5. No qual descreve o passo a passo deste trabalho.

Figura 5. Etapas do Desenvolvimento do Trabalho

3.1. Pesquisa bibliográfica

É um levantamento para obtenção de subsídios teóricos acerca do tema proposto. Elaborada a partir de material já publicado, constituído principalmente de: livros, revistas, publicações em periódicos e artigos científicos, jornais, monografias, dissertações, teses, internet, tudo isso com o objetivo de apresentar ao pesquisador o assunto pesquisado (PRODANOV; FREITAS, 2013). Essa fase realizou-se nas principais plataformas digitais como: *Google* Acadêmico, IEEE *Explorer* e *Elsevier* para verificar o atual estado da arte.

3.2. Construção

Consistiu na criação da arquitetura da solução; construção do robô e do Ator; criação de um ambiente de treinamento para o robô no *OpenAI Gym*; instalação e configuração do YOLOv4 para detecção de pessoa; instalação do Gazebo para simular o cenário 3D no qual o robô irá interagir.

3.3. Treinamento

Por meio da ferramenta Gym foi possível criar um ambiente de treinamento personalizado. O Gym está integrado ao ROS e ao Gazebo, onde são responsáveis por controlar o robô simulado, permitindo a troca de informações entre o ambiente de simulação e o algoritmo de aprendizado por reforço. Este ambiente atende a todos os requisitos exigidos pelo kit de ferramentas Gym para sua execução na biblioteca numérica *TensorFlow*.

4. Método Proposto

Para este trabalho foi considerado apenas uma pessoa para o robô seguir no simulador Gazebo, mas o YOLOv4 consegue detectar mais de uma pessoa ao mesmo tempo.

No processo de construção da solução foram definidos os seguintes itens:

- **Arquitetura:** identificação dos componentes da aplicação e construção da mesma;

- **Construção do Robô e do Ator:** o modelo do robô usado foi o *Pionner* 3-AT. Ele foi criado no formato de descrição de modelo, como o Formato de Descrição do Robô Unificado (URDF). O URDF é uma linguagem *Extensible Markup Language* (XML) usada para descrever a estrutura e as propriedades físicas de um robô. Para deixar com a aparência de um robô real, foi usado malhas (*meshes*) 3D, o termo malhas refere-se a modelos tridimensionais que representam a geometria de objetos. Já para o ator usou-se uma *skin* (pele) no formato COLLADA (.dae) do próprio Gazebo. A Figura 6 mostra como ambos ficaram.

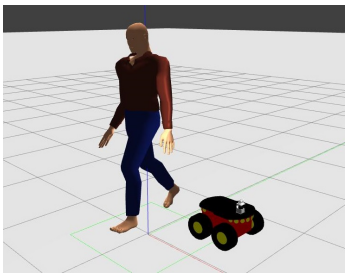

Figura 6. O Ator e o Robô

- **Ambiente de Treinamento:** o Gym foi usado na criação de um ambiente personalizado para o problema proposto. Isso envolveu a definição de estados, ações, recompensas que serão usados no processo de aprendizado por reforço.

- **YOLOv4:** é um modelo de detecção de objetos em tempo real que utiliza redes neurais convolucionais profundas. Para este trabalho foi usado o modelo para detecção de pessoas. Não houve necessidade de treinar um novo modelo, uma vez que o modelo usado disponível e treinado com o COCO *dataset* atendeu muito bem.

- **Gazebo:** foi utilizado para simular o comportamento do robô, facilitando o treinamento e validação da solução.

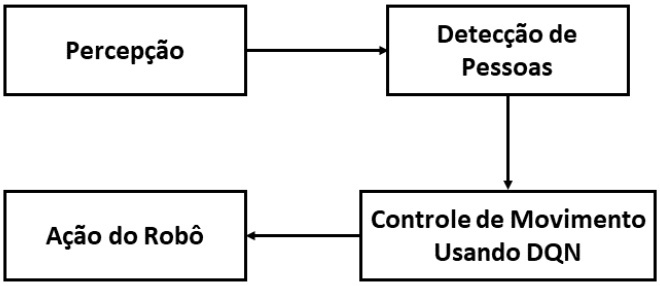

Figura 7. Estrutura do Método

A estrutura do método de rastreamento é mostrado na Figura 7. Para rastrear uma pessoa específica, é necessário identificá-la com base na visão do robô antes de iniciar o seu rastreamento e acompanhamento. O primeiro quadro o robô realiza a percepção do ambiente onde se encontra a pessoa por meio de uma câmera. Seguida pela detecção

da mesma pelo YOLOv4 onde o agente aprende a realizar o controle de movimento com base nos valores passados e posteriormente a ação do robô.

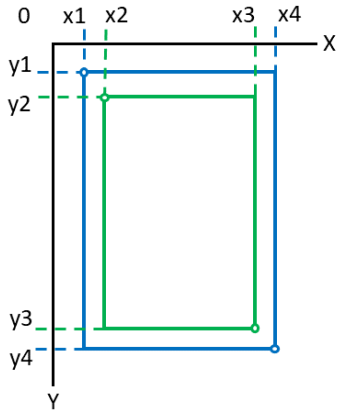

Figura 8. Retângulos

Na Figura 8, temos os retângulos usados como parâmetros e referências neste trabalho. O retângulo azul é o retângulo de referência que fica fixo na imagem da câmera do robô e o verde é o de detecção da pessoa. Através das diferenças das coordenadas superiores esquerda e inferiores direita desses dois retângulos é possível ao agente determinar a direção da pessoa detectada. É passada ao agente uma lista contendo o resultado dessa diferença de coordenadas onde pode ser calculada da seguinte forma:

- *left* = $x2 - x1$
- *right* = $x4 - x3$
- *top* = $y2 - y1$
- *bottom* = $y4 - y3$

Se o resultado dessas diferenças forem positivos ou zero a pessoa se encontra dentro do retângulo azul, caso contrário está fora do retângulo azul central. Fica a cargo da rede neural aprender por meios desses valores passados se a pessoa detectada está em movimento e com isso segui-la.

A Figura 9 mostra uma visão geral da abordagem proposta. Para treinar um robô seguidor de pessoas com aprendizado por reforço, utilizou-se de um algoritmo de aprendizado de máquina, o DQN para treiná-lo. O agente do DQN será responsável por receber as observações do ambiente, executando ações com base na política aprendida e ensinar a função de valor Q com base nas possíveis ações e selecionar a melhor para o robô. A função de valor Q na DQN desempenha um papel central ao estimar os valores de ação para guiar o agente na tomada de decisões, permitindo que ele aprenda uma política ótima através do treinamento por aprendizado por reforço.

Nesse caso, o estado seria a informação sobre a pessoa detectada, como sua posição em relação ao retângulo central. A ação seria a ação que o robô deve executar, como se mover em direção à pessoa, ele pode aprender a associar uma alta recompensa à ação de se mover em direção a uma pessoa. Através do treinamento e das atualizações da função de valor Q, o robô pode aprender a tomar ações mais eficientes e adequadas.

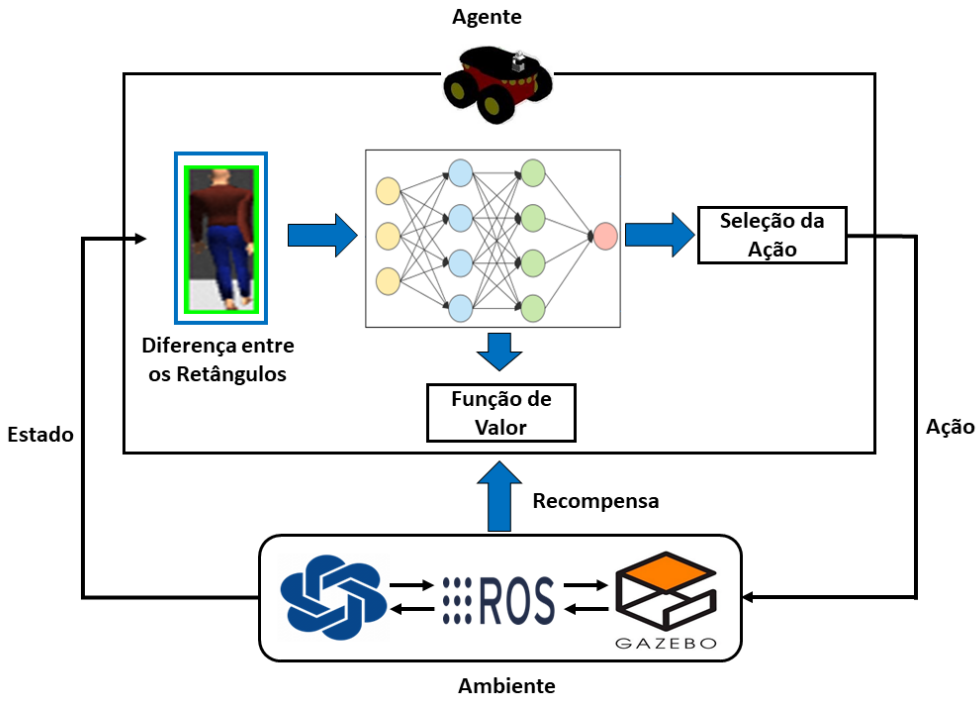

Figura 9. Arquitetura

O treinamento ocorreu no ambiente Gym da *OpenAI* que representa o cenário do robô onde o agente recebe informações sobre o estado do ambiente como as coordenadas dos retângulos que podem ser vistos na Figura 8. A função de recompensa fornece ao agente um *feedback* com base em seu desempenho ao seguir o alvo, podendo ser positiva quando o retângulo da pessoa detectada estiver dentro do retângulo central e negativa quando estiver fora do retângulo central. Essa interação do agente com o ambiente é armazenada, incluindo o estado atual, a ação tomada, a recompensa recebida e o próximo estado observado. Esses dados são utilizados para treinar a rede neural do DQN, atualizando seus pesos para maximizar as recompensas futuras.

4.1. Representação de Estados e Ações

O espaço de observação neste treinamento é baseado na detecção da pessoa por meio do componente de câmera no ambiente de simulação. A imagem da câmera possui um retângulo fixo e outro móvel que é o resultado da detecção pelo YOLOv4 quando há uma pessoa na cena. A diferença entre os dois retângulos representa o estado que é a entrada da rede neural do algoritmo DQN. O espaço de ações para o robô foi definido como velocidade linear no eixo X e velocidade angular no eixo Z.

4.2. Função de Recompensa

Em muitos problemas de Aprendizado por Reforço, identificar quando o objetivo ou alvo é alcançado pelo agente pode ser desafiador, pois o agente precisa ter acesso a informações detalhadas sobre o ambiente, como a localização precisa do alvo, características específicas relacionadas ao objetivo, ou qualquer outra informação relevante para identificar o sucesso da tarefa. Para este problema o agente robótico executa uma ação a_t em um estado s_t e recebe uma recompensa r_t representada pelo objetivo alcançado. Assim, o aprendizado do agente é direcionado pela função de recompensa R. Pose ser descrita da

seguinte forma: $R(s_t, a_t, s_{t+1})$, onde realiza uma ação a_t no estado atual s_t levando a um novo estado s_{t+1} pode ser descrita da seguinte forma:

A Equação 1, descreve as recompensas quando o agente alcança seu objetivo, recebendo uma recompensa de +1 (mantém o retângulo detectado da pessoa dentro do retângulo azul central). Recebe uma punição no valor 0 se o retângulo da pessoa detectada estiver fora do retângulo central. E recebe uma punição de -5 quando o retângulo da pessoa não for detectado pelo agente. O agente sabe quando a pessoa detecta estiver fora do retângulo central pela diferença dos cantos dos mesmos, essa diferença se dá pelas coordenadas dos cantos inferiores esquerdo e superiores direito dessas caixas delimitadoras.

$$R(s_t, a_t, s'_{t+1}) = \begin{cases} 1 & \textit{if isDentroRetan()} \\ 0 & \textit{if isForaRetan()} \\ -5 & \textit{if isPersonNoDetect()} \end{cases} \quad (1)$$

5. Resultados e Discussões

A Figura 10, mostra a pessoa distante do robô e na Figura 11 observa-se duas caixas delimitadora, sendo a maior, a azul fixa no meio da imagem e a verde é móvel à responsável pela detecção da pessoa. É por meio desses dois retângulos que o agente aprenderá. A Figura 12, mostra a pessoa perto demais do robô e por isso a caixa delimitadora que detecta a pessoa vista na Figura 13 acaba passando do tamanho da azul. Na Figura 14, podemos ver a distância encontrada pelo robô ideal que mantém a pessoa sempre dentro da caixa azul. O agente será o responsável por aprender qual a melhor distância deve ficar em relação a pessoa e em qual aceleração seria adequada para segui-la.

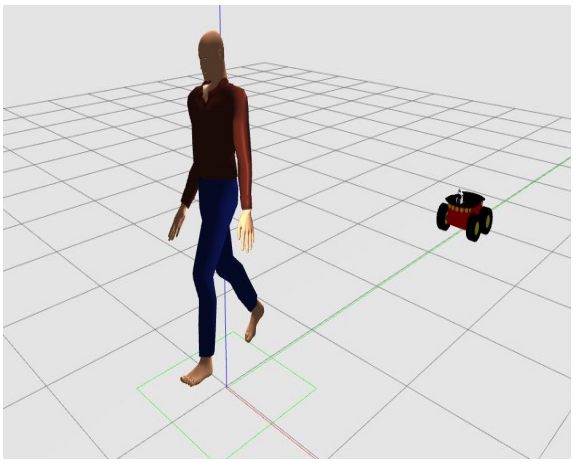

Figura 10. Robô Distante da Pessoa

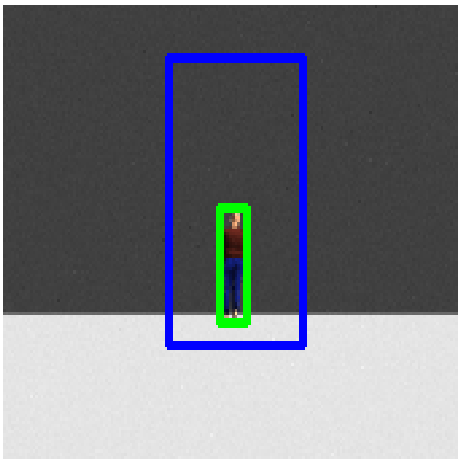

Figura 11. Perspectiva do Robô

O treinamento foi realizado utilizando os códigos gerados para simulação no ambiente Gazebo, Gym e ROS. Foi utilizado a biblioteca numérica *TensorFlow* para os cálculos e acompanhamento do resultado, que pode ser visto na Figura 16 onde temos no eixo Y as recompensas e no X os episódios. A passagem do tempo no simulador foi acelerada em 10x o tempo normal para este treinamento e foram realizados um pouco menos de 3.000 episódios. Apesar de poucos episódios treinados, o gráfico mostra que o agente aprendeu a pessoa detectada.

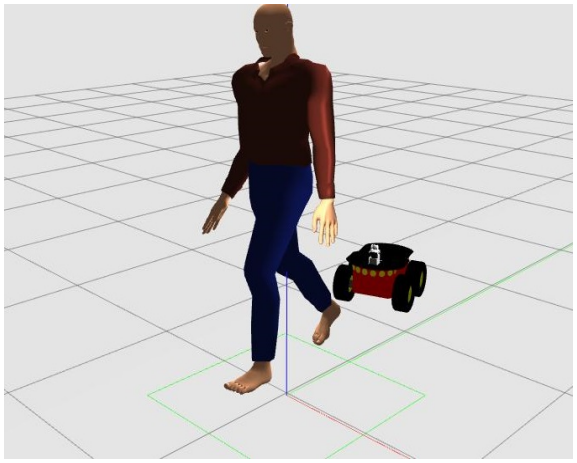

Figura 12. Perto Demais

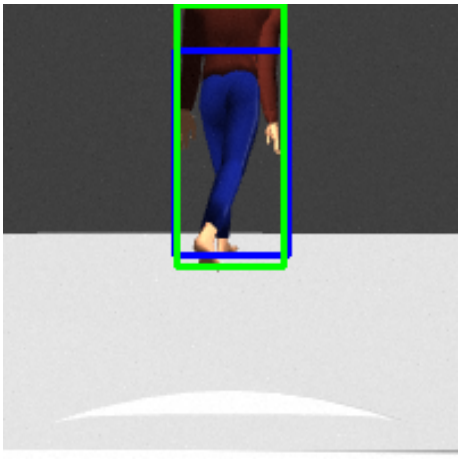

Figura 13. Perspectiva do Robô

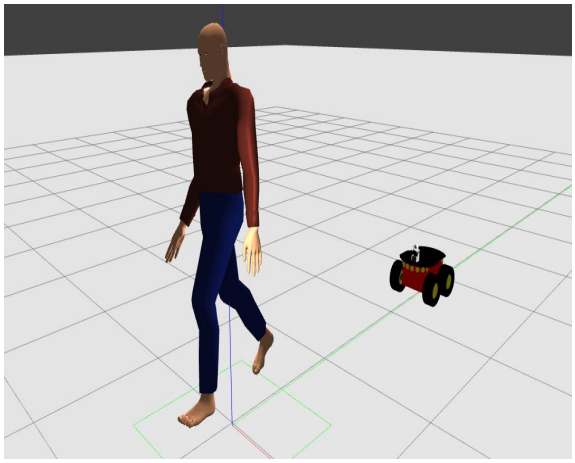

Figura 14. Distância Boa

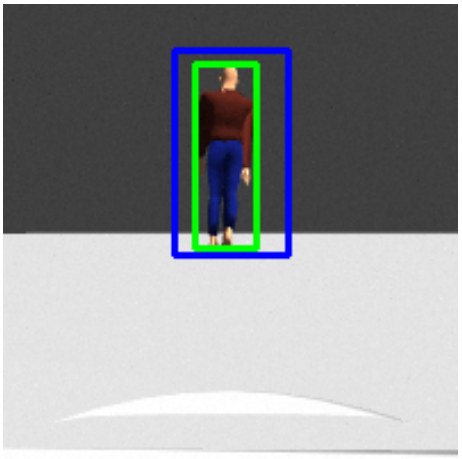

Figura 15. Perspectiva do Robô

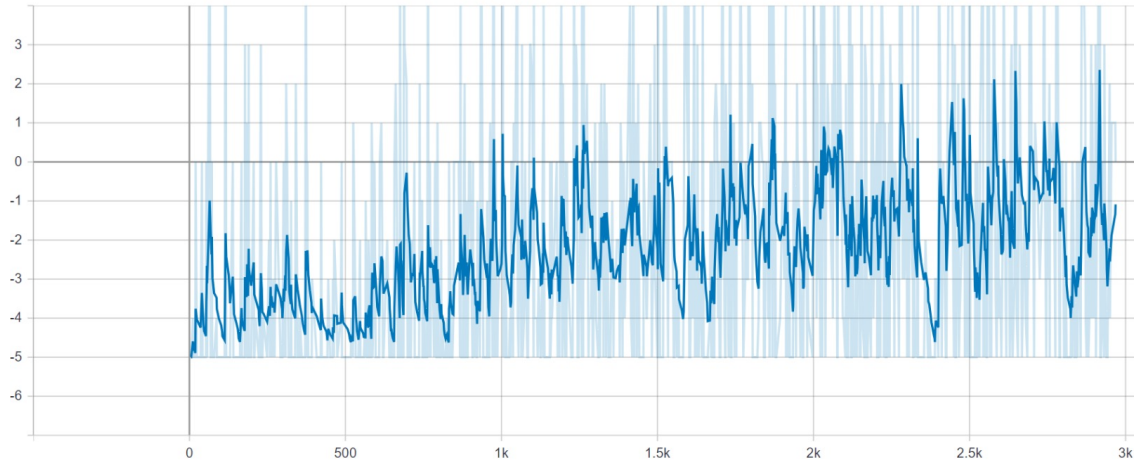

Figura 16. Total de Recompensas por Episódio

6. Conclusão

Este trabalho apresentou uma abordagem inovadora para o rastreamento e seguimento de pessoas por meio de robôs seguidores. Utilizou-se o aprendizado por reforço como método de treinamento, empregando o algoritmo *Deep Q-Network* (DQN). Os resultados obtidos são uma evidência da aplicabilidade do método proposto. O agente do DQN aprendeu a tomar decisões eficientes com base nas interações contínuas com o ambiente e na busca por recompensas positivas, mostrando um desempenho satisfatório no rastreamento e seguimento de pessoas.

A revisão bibliográfica realizada evidenciou a importância do aprendizado por reforço e das redes neurais profundas na área de robótica autônoma e destacou a relevância da abordagem adotada neste trabalho. A metodologia empregada, que envolveu pesquisa nas bases de dados, construção da solução e treinamento do modelo, mostrou-se adequada e permitiu alcançar resultados promissores.

As aplicações potenciais dessa técnica são diversas, abrangendo áreas como vigilância, busca e resgate, assistência humana e segurança. Essa abordagem permite que os robôs sigam e monitorem indivíduos de forma autônoma, facilitando o reconhecimento de padrões e comportamentos relevantes, bem como a realização de tarefas específicas com precisão e agilidade.

No entanto, ainda há espaço para melhorias e trabalhos futuros. É possível explorar diferentes arquiteturas de redes neurais e algoritmos de aprendizado por reforço para aprimorar ainda mais o desempenho do sistema. Além disso, é importante considerar a implementação prática do sistema em cenários reais e avaliar sua robustez e escalabilidade.

Referências

ALGABRI, R.; CHOI, M.-T. Deep-learning-based indoor human following of mobile robot using color feature. *Sensors*, MDPI, v. 20, n. 9, p. 2699, 2020.

ALVES, G. *Detecção de Objetos com YOLO – Uma abordagem moderna*. 2022. ⟨https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/?doing_wp_cron=1687179596.5247509479522705078125⟩. Acesso em 19 de junho de 2023.

ARAÚJO, A. M. et al. Detecção e destaque em vídeo de objetos utilizando yolo. Universidade Federal da Paraíba, 2022.

AYE, Y. Y. et al. A deep neural network based human following robot with fuzzy control. In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.: s.n.], 2019. p. 720–725.

BINGHAM, B. et al. Toward maritime robotic simulation in gazebo. In: IEEE. *OCEANS 2019 MTS/IEEE SEATTLE*. [S.l.], 2019. p. 1–10.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

BROCKMAN, G. et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

CHEN, B. X.; SAHDEV, R.; TSOTSOS, J. K. Integrating stereo vision with a cnn tracker for a person-following robot. In: SPRINGER. *Computer Vision Systems: 11th International Conference, ICVS 2017, Shenzhen, China, July 10-13, 2017, Revised Selected Papers 11*. [S.l.], 2017. p. 300–313.

HAN, D.; PENG, Y. Human-following of mobile robots based on object tracking and depth vision. In: *2020 3rd International Conference on Mechatronics, Robotics and Automation (ICMRA)*. [S.l.: s.n.], 2020. p. 105–109.

HU, R.; ZHANG, Y.; LI, C. Toward stable astronaut following of extravehicular activity assistant robots using deep reinforcement learning. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 19, n. 3, p. 17298806221108606, 2022.

JOSEPH, L. *Robot operating system (ros) for absolute beginners*. [S.l.]: Springer, 2018.

KOIDE, K.; MIURA, J. Identification of a specific person using color, height, and gait features for a person following robot. *Robotics and Autonomous Systems*, Elsevier, v. 84, p. 76–87, 2016.

KULKARNI, J.; PANTAWANE, P. Person following robot based on real time single object tracking and rgb-d image. In: IEEE. *2022 International Conference on Signal and Information Processing (IConSIP)*. [S.l.], 2022. p. 1–5.

LUU, Q. T. *Q-Learning vs. Deep Q-Learning vs. Deep Q-Network*. 2023. Disponível em: ⟨https://www.baeldung.com/cs/q-learning-vs-deep-q-learning-vs-deep-q-network⟩. Acesso em 30 de junho de 2023.

MNIH, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

MOREIRA, V. R. F. *Controle inteligente de um robô móvel omnidirecional com tomada de decisão utilizando aprendizagem por reforço*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2021.

Open Source Robotics Foundation. *Gazebo: Simulação de Robô Facilitadal*. 2023. ⟨https://classic.gazebosim.org/⟩. Acesso em 1 de junho de 2023.

OPENAI. *OpenAI Gym Beta*. 2023. Disponível em: ⟨https://openai.com/research/openai-gym-beta⟩. Acesso em 5 de junho de 2023.

PRODANOV, C. C.; FREITAS, E. C. D. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. [S.l.]: Editora Feevale, 2013.

QIAN, W. et al. Manipulation task simulation using ros and gazebo. In: *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. [S.l.: s.n.], 2014. p. 2594–2598.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

ROS. *The ROS Ecosystem: Software, Hardware, Documentation, and People*. 2023. Disponível em: ⟨https://www.ros.org/blog/ecosystem/⟩. Acesso em 5 de junho de 2023.

ROS. *Why ROS?* 2023. Disponível em: ⟨https://www.ros.org/blog/why-ros/⟩. Acesso em 1 de junho de 2023.

SCHLEGEL, C. et al. Vision based person tracking with a mobile robot. In: CITESEER. *BMVC*. [S.l.], 1998. p. 1–10.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

TAROKH, M.; MERLOTI, P. Vision-based robotic person following under light variations and difficult walking maneuvers. *Journal of Field Robotics*, Wiley Online Library, v. 27, n. 4, p. 387–398, 2010.

VASILEV, I. et al. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. [S.l.]: Packt Publishing Ltd, 2019.

WATKINS, C. J.; DAYAN, P. Q-learning. *Machine learning*, Springer, v. 8, p. 279–292, 1992.

WATKINS, C. J. C. H. Learning from delayed rewards. King's College, Cambridge United Kingdom, 1989.

WELTER, A. R. *Aprendizado por reforço profundo para navegação de um veículo guiado automaticamente*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2022.

WIERING, M. A.; OTTERLO, M. V. Reinforcement learning. *Adaptation, learning, and optimization*, Springer, v. 12, n. 3, p. 729, 2012.

YOSHIMI, T. et al. Development of a person following robot with vision based target detection. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2006. p. 5286–5291.

ZAMORA, I. et al. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.

ZHANG, M. et al. Vision-based target-following guider for mobile robot. *IEEE Transactions on Industrial Electronics*, v. 66, n. 12, p. 9360–9371, 2019.