Simulação de Veículos Autônomos em Ambientes 2D Usando Aprendizado por Reforço Aplicados para a Solução de Labirintos

Darlan Porsch*, João Antônio Oldenburg Vieira*, Ênio dos Santos Silva*
*GPEB – Grupo de Pesquisas em Engenharia Biomédica
*Curso de Engenharia Elétrica
*Instituto Federal de Santa Catarina (IFSC)
Av. Abrahão João Francisco, 3899, Bairro Ressacada
CEP: 88307-303, Itajaí, Santa Catarina, Brasil
E-mails: darlanporsch11@gmail.com, joao.oldenburg@gmail.com, enio.silva@ifsc.edu.br

***Resumo*—Este trabalho apresenta uma discussão sobre o uso de inteligência artificial (*artificial intelligence* - IA) e aprendizado de máquina (*machine learning* - ML) aplicados à navegação de veículos autônomos. Particularmente, visando a investigação de técnicas de aprendizado por reforço (*reinforcement learning* - RL), a implementação do algoritmo *Q-learning* é considerada para a solução de problemas de navegação representados por labirintos. Adicionalmente, a fim de obter alternativas para a construção e visualização de diferentes labirintos, assim como possibilitar o acompanhamento visual das tomadas de decisão de agentes robóticos simulados durante as etapas de treinamento e teste, uma interface de desenvolvimento de RL (RL *framework*) é aqui concebida. Resultados de simulação numérica são apresentados e permitem inferir acerca da qualidade das soluções de navegação realizadas por meio das estratégias de ML aqui investigadas, confirmando a eficácia das implementações de RL e do RL *framework* desenvolvido neste trabalho de pesquisa.**

***Index Terms*—Ambientes virtuais, aprendizado de máquina, aprendizado por reforço, inteligência artificial, navegação inteligente, *Q-learning*, veículos autônomos.**

I. Introdução

Em [1], segundo o Índice Global 2022 de Adoção de Inteligência Artificial (*artificial intelligence* - AI), companhias de todo o mundo vêm adotando, de maneira consistente, o uso de AI em seus modelos de negócios. Nesse cenário, o interesse da indústria por serviços de tecnologias envolvendo aprendizado de máquina (*machine learning* - ML) e AI vem motivando a comunidade acadêmica para a formação de recursos humanos em AI e ML aplicados em diferentes problemas do mundo real [2]. Para a área da robótica, a AI consiste na investigação de agentes inteligentes que recebem sinais de percepção de um ambiente e executam determinadas ações, isto é, implementam funções que mapeiam sequências de percepções em ações [3]. Nesse contexto, as áreas de AI relacionadas a aplicações e desenvolvimento de veículos autônomos vêm ganhando um importante destaque na industria e na literatura do estado-da-arte em AI e ML [4], [5]. Dentre toda a gama de aplicações, os veículos autônomos podem exercer funções desde a limpeza de determinada área e controle de estoque, até a navegação de agentes robóticos que exploram ambientes nocivos aos seres humanos. Em ambientes nocivos como minas, estruturas físicas (construções) em situação de desabamento, ambientes com elevadas taxas de toxicidade, entre outros, a capacidade de adaptação do agente robótico (explorador do ambiente) é fundamental para o mapeamento dinâmico das regiões perigosas, assim como a localização das saídas a partir de qualquer local explorado. Nesses casos, geralmente o problema de navegação consiste no resgate de pessoas que se encontram em situação de risco causada pelos ambientes supracitados [6].

Independente dos cenários e aplicações considerados, os veículos autônomos são constituídos por diversos sensores empregados como ferramentas para a aquisição de sinais de entrada utilizados pelos sistemas de AI e ML. Especificamente, as leituras dos sensores são consideradas para a determinação de ações (tomada de decisão) sequenciais com base no histórico de sinais adquiridos em tempo real.

Dessa forma, buscando a solução de problemas de navegação e visando simular a complexidade dos ambientes supramencionados, a solução de labirintos é considerada como objeto de estudo neste trabalho. Especificamente, agentes robóticos equipados com sensores de presença (a saber: sensores laterais direito e esquerdo; e sensor frontal) são simulados e inseridos em labirintos bidimensionais (2D) com opção de visualização tridimensional (3D). Na literatura, a solução de labirintos pode ser obtida através de técnicas associadas à solução de problemas denominados *grid world* e do caminho mais curto [7], [8]. Particularmente, a busca por soluções inteligentes para o problema do labirinto ainda é um tópico ativo de pesquisa do estado da arte [6], [9].

Atualmente, o estado da arte na área de veículos autônomos investiga tecnologias destinadas à estimação de séries temporais e classificação de processos estocásticos, onde a tomada de decisão é realizada com base no histórico de um dado sinal ou sistema (processo de decisão de Markov) [10]–[12]. Considerando o estado da arte da área, tais tecnologias empregam soluções usando aprendizado por reforço (*reinforcement learning* - RL) [12], [13]. Nesse contexto, a abordagem de RL com *Q-learning* tem sido de grande valia para a solução de

problemas de navegação, pois tal abordagem permite que um agente aprenda a tomar decisões "ótimas" a partir de regiões desconhecidas, através do uso de fatores de recompensas e penalidades, os quais dependem da escolha da ação a ser tomada pelo agente. Em resumo, RL com *Q-learning* é uma subárea de ML e AI que dedica atenção especial ao treinamento de modelos (denominados agentes) para, a partir de tomadas de decisões sequenciais, alcançarem um objetivo (destino) específico dentro de um ambiente (cenário) arbitrário [10]–[12].

Portanto, visando a elaboração e investigação de uma navegação inteligente com gestão de auto-localização, planejamento de rotas e construção de mapas de direções (mapas de ações a serem tomadas dada uma certa localização), este trabalho investigada tecnologias de AI e ML (RL com *Q-learning*) aplicadas em simulações de veículos (robôs) autônomos destinados a solução de labirintos. Particularmente, este trabalho utiliza labirintos 2D e os apresenta como ambientes que podem ser visualizados e/ou implementados tanto em 2D quanto em 3D.

II. Fundamentação Teórica

Em resumo, o objetivo principal de um agente autônomo (robô) é construir um mapa e usá-lo para se localizar no ambiente em que está inserido, conhecido como mapeamento e localização simultâneos (*simultaneous localization and mapping* - SLAM) [14]. Nesse contexto, o planejamento aprimorado das trajetórias do robô torna-se fundamental. Dessa forma, nesta seção é apresentada uma breve revisão do processo de decisão de Markov e da equação responsável pelo treinamento e atualização das funções *Q-learning*, que norteiam as decisões tomadas pelos agentes autônomos.

A. Processo de Decisão de Markov

Particularmente, a tecnologia de aprendizado por reforço (RL) é formalmente descrita por um processo de decisão de Markov (*Markov decision process* - MDP). Tipicamente, sistemas usando RL são compostos por agentes que atuam de forma dinâmica na exploração de determinados ambientes. Especificamente, a cada iteração (passo) realizada pelo agente, um novo estado s é assumido. Nesse estado s, o agente executa apenas as ações a permitidas no ambiente. Neste trabalho, os ambientes propostos são labirintos 2D que permitem as ações de navegação correspondentes às locomoções para "cima, baixo, direita ou esquerda". Considerando um processo de decisão de Markov de 1ª ordem, a ação a escolhida pelo agente depende exclusivamente do estado atual s_t, já que o processo de decisão considera que toda a informação relevante do passado $(t-1, t-2, t-3, \ldots)$ está contida no estado atual s_t [15].

Para a solução de labirintos, cada local em que o robô se encontra é considerado como um estado s. Em cada estado s é associado um valor de recompensa r que é diretamente relacionado com a obtenção do objetivo final do robô (que nesse caso é encontrar a saída do labirinto). Portanto, um MDP é um processo de recompensa de Markov dependente de decisões. Assim, todos os estados são Markovianos e a navegação no labirinto é dividida em iterações (ou passos). A cada passo, um estado s é assumido e uma nova ação a é tomada considerando apenas o estado atual s_t. Matematicamente, um MDP é definido como uma tupla (S, A, T, R, γ). Assim como em [16], aqui S, A, T, R e γ são definidos como:

- $\mathbf{S}$ é o conjunto de estados finitos. Cada posição do labirinto representa um estado diferente;
- $\mathbf{A}$ é o conjunto de ações finitas. Para ocorrer a mudança de estado, o robô deve realizar uma ação de locomoção, por exemplo, ir para a esquerda, frente ou direita;
- $\mathbf{T}$ é a matriz de probabilidades de transição de estados $T(s_{t+1}|s_t, a_t)$. Probabilidade de ir à um estado futuro s_{t+1}, dado que no estado atual s_t será realizada a ação a_t;
- $\mathbf{R}$ é a função de recompensas que atribui o valor da recompensa fornecida ao robô quando o estado s é alcançado, $R(s, a) = E[r|s, a]$;
- γ é um fator de desconto ($\gamma \in [0, 1]$) que indica a importância das recompensas em cada estado ao longo de um percurso.

Então, para a solução de labirintos contendo estados finitos (posições possíveis do agente no labirinto) e múltiplas ações, o MDP visa aprender uma política π de decisão que eleve ao máximo o desempenho de uma dada tarefa ao longo do tempo. Nesse contexto, o objetivo do agente é percorrer o ambiente e obter o máximo de recompensas (e/ou o mínimo de penalidades) possíveis até a chegada ao seu destino final. Para isso, tal agente deve aprender uma política π que determine a ação a "ótima" a ser tomada em cada estado s_t que o levará ao seu estado futuro s_{t+1} (posição futura no labirinto), sendo o estado atual s_t estatisticamente suficiente para a estimação (previsão) do estado futuro s_{t+1} [15]. Uma política π pode ser avaliada pela soma total das recompensas de um estado inicial s_0 até o estado atual s_t, como mostrado por $V^\pi(s)$ em (1) (onde γ representa um fator de ponderação aplicado nas recompensas r_{t-i} correspondentes a cada estado s_{t-i}, $\forall i \in \mathbb{Z}$).

$$\begin{aligned} V^\pi(s) &= r_t + \gamma r_{t-1} + \ldots = E\left\{\sum_{i=0}^{\inf} \gamma^i r_{t-i} | s_0 = s, \pi\right\} \\ V^\pi(s) &= \sum_{a_t} \pi(s_t, a_t) \sum_{s_t, a_t, s_{t+1}} [R(s_t, a_t, s_{t+1}) + \gamma V^\pi(s_{t+1})] \end{aligned} \tag{1}$$

Adicionalmente, a função

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) V^*(s_{t+1}),$$

é definida para obter o valor atribuído a escolha da ação a no estado s, seguindo a política ótima π^*. Assim, em RL, a estimação da função Q^* consiste no aprendizado de uma solução de um MDP [15].

B. Equação de Atualização da Função Q-Learning

A fim de evitar uma abordagem gananciosa, na qual o agente busca recompensas imediatas em detrimento das recompensas globais obtidas ao longo do percurso, esta seção apresenta o

método de RL com *Q-learning*. Assim, o algoritmo *Q-learning* é adotado para aprendizado iterativo, estimando os valores da função Q para cada par estado-ação no MDP, com base nas recompensas dos estados subsequentes, de acordo com a equação de *Bellman* dada em (2) [15]:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_{t+1} + \gamma \arg\max_a Q_t(s_{t+1}, a_{t+1})]$$

$$Q(s, a) = Q(s, a) + \alpha([R(s, a, s') + \gamma \arg\max_i Q(s', a_i)]) \quad (2)$$

Onde: α é a taxa de aprendizado; $Q(s, a)$ é o valor Q da ação a no estado s; e $Q(s', a_i)$ é o valor Q da ação a_i no estado s'.

Essa equação expressa que o valor Q de uma ação a no estado s é atualizado pela diferença entre a recompensa imediata obtida ao tomar a ação a, e chegar no estado s', somada ao valor máximo dos valores Q dos estados subsequentes, ponderados pelo fator de desconto γ. A partir desse procedimento, os valores da função *Q-learning* convergem para valores "ótimos", resultando na politica π "ótima" de tomada de decisão dentro do MDP.

III. Desenvolvimento de labirintos

Visando a simulação de agentes robóticos explorando ambientes de navegação, neste trabalho foram desenvolvidos códigos para a construção de labirintos (com graus pré definidos de aleatoriedade). Aqui, os labirintos considerados são representados por matrizes de dimensão $N \times M$, sendo as paredes dos labirintos definidas como 0 e os caminhos livres como 1. Especificamente, o processo de construção do labirinto inicia a partir de uma matriz de zeros, isto é, formado apenas por paredes. Em seguida, os locais de entrada (estado inicial) e de saída do labirinto (estado final) são definidos e, fazendo uso do método de busca em profundidade (originalmente apresentado em [17]), o caminho entre a entrada e a saída do labirinto é construído.Tal construção consiste em, a partir do estado inicial, sendo este 0, converter os seus correspondentes locais adjacentes para 1. Esse procedimento representa a construção de um espaço livre "1" através da "destruição" de uma parede "0". Dessa forma, as próximas "paredes" a serem "quebradas" são determinadas e, assim, de passo em passo, os caminhos do labirinto são construídos. Particularmente, cada elemento da matriz do labirinto é visitado apenas uma única vez. Inicialmente, essa estratégia de construção resulta em labirintos que apresentam apenas um único caminho possível entre a entrada e a saída, porém com diversos outros caminhos adjacentes que não são interligados.

No entanto, para problemas de navegação de veículos autônomos, é interessante que o labirinto apresente no mínimo dois caminhos possíveis entre a entrada e a saída. Então, após a construção da versão preliminar do labirinto (versão com apenas um caminho entre a entrada e a saída), locais aleatórios do labirinto (elementos da matriz $N \times M$) são novamente "visitados" a fim de encontrar pontos interessantes para a construção de ligações (através da quebra de paredes adjacentes) entre os diferentes caminhos do labirinto. A Figura 1 ilustra um labirinto de dimensão 15×15 construído pelo algoritmo desenvolvido neste trabalho.

A partir da Figura 1, nota-se que que todas as bordas do labirinto são geradas de forma que sejam compostas por paredes, ou seja impossibilitam que o agente saia do labirinto antes de alcançar o seu objetivo final. A célula em azul (canto superior esquerdo) representa a posição inicial de treinamento do agente e a célula em verde (canto superior direito) representa o objetivo do agente, isto é, o local que é considerado como a saída do labirinto.

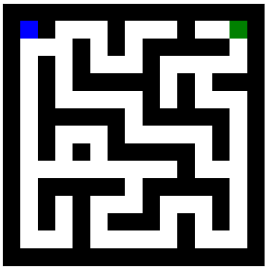

Figura 1. Exemplo de labirinto construído a partir do algoritmo aqui desenvolvido.

IV. Etapa de Treinamento

Para a locomoção do agente autônomo no labirinto, um conjunto de sensores de navegação (sensores de presença) é considerado. Especificamente, o agente é equipado com sensores laterias [esquerda (E) e direita (D)] e frontal (F). Dessa forma, um vetor [E, F, D] informa 1 para as posições livres e 0 para as posições com obstáculos (paredes). Durante a locomoção do agente, as ações que resultarem em colisões com as paredes dos labirintos serão desconsideradas pelo modelo de RL implementado, pois o agente deve se locomover apenas pelos caminhos disponíveis dentro do labirinto.

Dado o estado atual s_t do agente, as ações a possíveis em s_t são definidas. Em seguida, ou o agente recebe recompensas negativas (penalidades, -100) caso a ação a tomada não alcance imediatamente o objetivo final, ou uma recompensa positiva (1000) caso alcance imediatamente o objetivo final (saída do labirinto). Especificamente, as recompensas foram configuradas por meio de testes empíricos, com o propósito de aprimorar o desempenho do agente. Durante esse processo, ficou claro que a discrepância entre as recompensas e penalidades desempenhou um papel vital na otimização do aprendizado. Consequentemente, optou-se pelos valores -100 e 1000 devido a otimização alcançada e ao seu notável impacto visual. Nesse caso, a consideração de penalidades visa "estimular" o agente a encontrar (aprender) o menor percurso até a saída do labirinto (caminho ótimo).

Conforme discutido na Seção II, o agente autônomo se movimenta por meio de ações estimadas a partir de valores da função *Q-learning*. Aqui, essa função é representada por uma matriz Q, onde as colunas de Q correspondem às ações (cima, baixo, direita, esquerda) disponíveis em cada estado; e o número de linhas de Q corresponde ao número de elementos

do labirinto, isto é, o número de estados do labirinto. Por exemplo, no labirinto ilustrado na Figura 1, como a dimensão é 15×15, a matriz Q possui um total de 225 estados (linhas). Dessa forma, o agente percorre o labirinto e, a medida que realiza as possíveis ações, a matriz Q é atualizada. Inicialmente a matriz Q é composta apenas por zeros e, ao longo da caminhada do agente, novos valores vão sendo definidos para a matriz Q, considerando as correspondentes recompensas de cada estado e também a equação de atualização expressa pela equação de *Bellman* (para mais detalhes, veja Seção II). Nesse contexto, um ciclo de treinamento (denominada época) é concluído quando o agente alcança o seu objetivo final (sair do labirinto). A partir de então, um novo ciclo (época) de treinamento é iniciado. Particularmente, o número de épocas necessárias para que ocorra a convergência do treinamento varia de acordo com as dimensões da matriz Q. O quadro abaixo apresenta o algoritmo *Q-learning* aqui implementado.

Algoritmo 1: Q-LEARNIG

para *épocas* $= 1$ *até* N_{epocas} **faça**
  O estado atual $s_t \leftarrow$ estrada do labirinto
  enquanto estado atual $\neq$ estado final **faça**
    $a_t \leftarrow$ a ação tomada $\arg\max_{a_t}\{Q_t(s_t, a_t)\}$
    se número aleatório $\leq$ taxa de exploração
      $a_t \leftarrow$ ação aleatória
    fim
    Executa a ação a_t e obtém o estado futuro s_{t+1}
    se estado futuro s_{t+1} = estado final
      obtém a recompensa imediata r_{t+1} = 1000
    senão
      obtém a recompensa imediata r_{t+1} = -100
    fim
    Atualiza Q: $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_{t+1} + \gamma \arg\max_{a_{t+1}}\{Q_t(s_{t+1}, a_{t+1})\}]$
    Atualiza o estado atual: $s_t = s_{t+1}$
  fim
fim

Durante a etapa de treinamento, é possível observar a evolução gradual da política π "aprendida" pelo agente, assim como observar as alterações (atualizações) realizadas na matriz Q a cada época de treinamento. Como mencionado na Seção II, o uso de uma política gananciosa não é desejado. Portanto, uma taxa de exploração aleatória é considerada durante o aprendizado do agente autônomo. A taxa de exploração define a probabilidade em que o agente realiza tomadas de decisão aleatórias (sem seguir a orientação da matriz Q). A tomada de decisões aleatórias permite que o agente realize uma maior exploração do labirinto e, consequentemente, que aprimore a sua política π nos estados que não tenham sido visitados com frequência.

Assim como o valor da taxa de exploração, geralmente os valores da taxa de aprendizado α e do fator de desconto γ, dependem da dimensão do labirinto e da sua complexidade. Tais valores são denominados como hiper-parâmetros do modelo de RL e são empiricamente determinados. Nesse contexto, a taxa de aprendizado α corresponde a velocidade com que o agente atualiza a matriz Q. Valores elevados de α fazem com que o agente aprenda rapidamente e se adapte às recompensas obtidas de imediato, entretanto isso pode causar dificuldades de convergência. Por outro lado, baixos valores de α fazem com que o agente aprenda de forma mais lenta, dando mais importância às informações de longo prazo, e, assim, aumentando as chances de convergência do modelo de RL (aqui, levando em conta os pontos supramencionados, foi utilizado α = 1 em todos os treinamentos). Já o fator de desconto γ determina a importância que será atribuída às recompensas futuras em relação às recompensas que são concedidas de imediato. Valores baixos de γ darão mais importância para as recompensas imediatas, dando menos importância para os resultados obtidos a longo prazo. Valores baixos de γ são indicados nos casos em que o ambiente sofre mudanças constantes. Nesse caso, o agente deve "aprender" rapidamente. Por outro lado, valores elevados de γ fazem com que o agente priorize recompensas que serão mais benéficas a longo prazo. Valores elevados de γ são indicados nos casos em que o ambiente não sofre alterações constantes (ambientes estáveis). Nesse caso, o agente pode considerar um aprendizado a longo prazo.

V. Discussão e Análise de Resultados

Neste trabalho, as simulações computacionais foram realizadas usando a linguagem de programação Python e adotando, principalmente, as bibliotecas Numpy, Matplotlib, Random, Time e Socket. Além disso, foram feitas implementações utilizando o motor de jogos Unity3D, assim como usando a linguagem C# (considerando as bibliotecas System e UnityEngine) para realizar a comunicação entre o Python e o Unity3D. Para executar os experimentos em Python, foi utilizado o serviço de computação remota "Google Colaboratory", o qual oferece uma máquina virtual Debian Linux com 4 CPUs Intel Xeon de 2,20 GHz, 12,7 GB de RAM e uma unidade de processamento gráfico Nvidia Tesla P100 com 16 GB de RAM.

A. Considerações Sobre a Taxa de Exploração

Conforme discutido nas seções anteriores, como o ambiente simulado corresponde a um labirinto estático e o objetivo principal é encontrar a saída desse labirinto, o fator de desconto é definido como $\gamma = 0,70$ e as taxas de exploração (TE) consideradas para investigação são de 10 e 70%. A Figura 2 apresenta o labirinto de dimensão 21×21 construído aqui para o treinamento dos agentes autônomos usando RL.

A partir do labirinto apresentado na Figura 2, os agentes autônomos com $\gamma = 0,7$ e TE de 0,1 e 0,7 são treinados por 1000 épocas usando o algoritmo *Q-learning*.

Após a conclusão do treinamento e usando mapas de calor, as Figuras 3 e 4 mostram as movimentações dos agentes durante a etapa de treinamento.

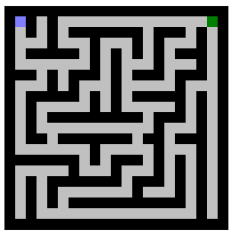

Figura 2. Labirinto de dimensão 21×21 utilizado no treinamento de RL.

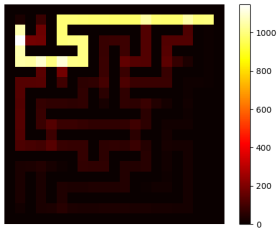

Figura 3. Mapa de calor para o agente com $TE = 0,1$.

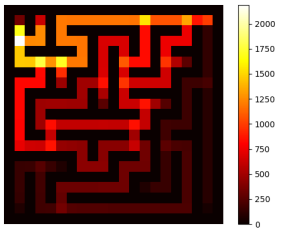

Figura 4. Mapa de calor para o agente com $TE = 0,7$.

Das Figuras 3 e 4, observa-se que o mapa de calor correspondente ao treinamento do agente definido com $TE = 0,7$ apresentou uma exploração mais abrangente do labirinto. No entanto, essa maior abrangência da área do labirinto torna o treinamento desse agente mais lento quando comparado com o treinamento do agente com $TE = 0,1$. A Tabela I apresenta os resultados obtidos pelos dois agentes supracitados.

Tabela I
RESULTADOS OBTIDOS PELOS AGENTES USANDO RL.

TE	Tempo (segundos)	Passos Época 1	Passos Época 1000
0,1	106,1123	354	30
0,7	296,0654	336	30

Como mostrado na Tabela I, ambos os agentes convergem para o caminho ótimo de 30 passos até a saída do labirinto após a realização de 1000 épocas de treinamento. Além disso, observa-se que esses agentes obtiveram aproximadamente o mesmo número de passos para encontrar a solução do labirinto na primeira época. No entanto, o aspecto mais contrastante é o tempo empregado para a realização do treinamento completo. Nesse aspecto, o agente com $TE = 0,7$ obteve um gasto computacional maior, o qual necessitou de um tempo de treinamento aproximadamente $2,79$ vezes maior quando comparado com o agente com $TE = 0,1$. A Figura 5 apresenta a solução ótima do labirinto encontrada (aprendida) pelos dois agentes.

Vale ressaltar que mesmo considerando valores baixos para TE, o agente é capaz de solucionar o labirinto. Entretanto, a política π (de tomadas de decisão simbolizadas aqui por setas orientadas para cima, baixo, direita ou esquerda) calculada durante o MDP é severamente prejudicada e a solução obtida pode não ser a solução ótima do labirinto. Nesse contexto, embora agentes diferentes sejam definidos com taxas de exploração diferentes, os caminhos obtidos podem convergir

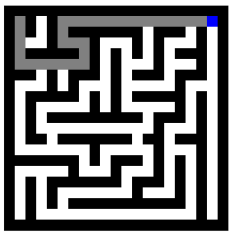

Figura 5. Percurso final aprendido pelos agentes autônomos.

de maneira parecida quando labirintos de pequenas dimensões são considerados. Por outro lado, como apresentado pelas Figuras 6 e 7, as políticas gerais π calculadas pelos agentes são visivelmente distintas. Nesse caso, a política π obtida pelo agente com $TE = 0,1$ apresenta um desempenho inferior quando comparada com a política do agente com $TE = 0,7$.

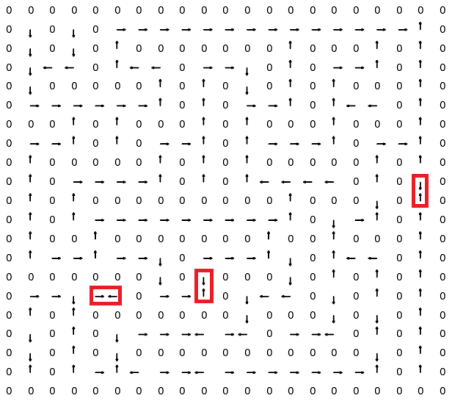

Figura 6. Mapa correspondente a política π de tomada de decisão para o agente com $TE = 0,1$.

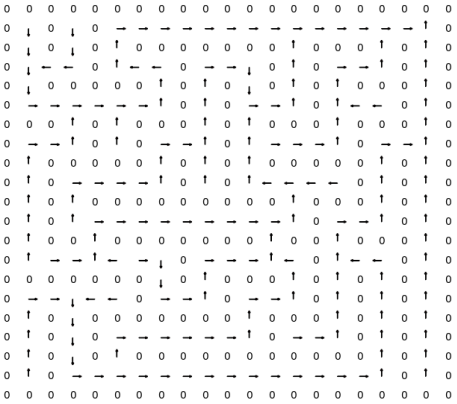

Figura 7. Mapa correspondente a política π de tomada de decisão para o agente com $TE = 0,7$.

A partir das Figuras 6 e 7, ao analisar o mapa de política de ambos os agentes, é possível observar algumas falhas na política calculada pelo agente com $TE = 0,1$. Com a finalidade de facilitar a visualização e localização no labirinto

em questão, algumas das políticas são destacadas em vermelho na Figura 6. Nesses regiões destacadas ocorrem laços onde o agente é repetidamente orientado a seguir a direção oposta a sua direção atual, resultando em um *loop* na política π. Nesse caso, a política é ótima apenas no caminho correspondente a solução ótima do labirinto. Em contrapartida, ao observar a política π calculada pelo agente com $TE = 0,7$, nota-se que essa política é ótima para qualquer localização do labirinto. Assim, assumindo a maior taxa de exploração, o agente é capaz de encontrar a saída a partir de qualquer local do labirinto.

B. Considerações Sobre a Confiança na Tomada de Decisões

Tendo em vista a avaliação do desempenho do agente para a solução de labirintos, a métrica aqui denominada "fator de confiança" é considerada. Essa métrica é obtida através do valor quadrático médio normalizado de cada ação tomada em um determinado estado da matriz Q, ou seja, o "fator de confiança" representa o valor correspondente da melhor ação a ser tomada por um agente em um dado estado, de acordo com os valores da matriz Q. Considerando apenas o caminho ótimo correspondente à solução do labirinto aprendida pelo agente, a métrica indica o nível de certeza que o agente possui quanto ao caminho ótimo obtido (aprendido) por ele para alcançar a saída do labirinto. Complementando a informação proporcionada pela política π sobre qual a ação é tomada em cada local do labirinto, este indicador proporciona a informação sobre o grau de confiança do agente na ação indicada pela política π. Assim, tal medida é de grande valia para a análise das soluções de labirintos durante o percurso ótimo obtido.

A Figura 8 mostra um labirinto de dimensão 25×25 no qual um agente com $TE = 0,50$ e $\gamma = 0,90$ é treinado por 700 épocas. A Figura 9 apresenta o mapa de calor dos "fatores de confiança" correspondentes ao caminho ótimo obtido após o treinamento.

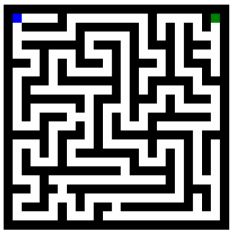

Figura 8. Labirinto 25×25 utilizado para treinamento.

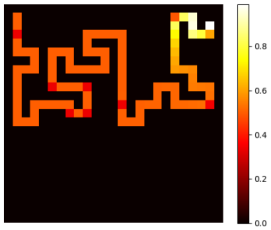

Figura 9. Mapa de calor do fator de confiança calculado sobre o caminho ótimo do labirinto.

Da Figura 9, nota-se que, durante o treinamento, o agente converge para o melhor caminho de solução do labirinto, adquirindo a "confiança" média de $52,31\%$ (equivalente a média de todos os fatores de confiança calculados ao longo do caminho ótimo). Particularmente, os locais com coloração mais escura (na Figura 9) representam uma maior dúvida do agente quanto às tomadas de decisão, enquanto os locais com coloração mais clara representam uma maior confiança. Nesse contexto, é possível observar que que a medida que o agente se aproxima da saída do labirinto, a confiança na tomada de decisão aumenta substancialmente. Por outro lado, na entrada do labirinto o agente apresenta valores mais baixos da taxa de confiança.

C. Análise Comparativa da Solução de Labirintos com Diferentes Dimensões

Nesta seção, visando a comparação da solução de labirintos com diferentes dimensões, as seguintes métricas foram consideradas: tempo de execução, "taxa de confiança" sobre o caminho ótimo, número de passos efetuados para a solução do labirinto segundo a política final π obtida na última época (Pπ); e o número de passos ideais necessários para a solução do labirinto (PI). Para essas simulações, foram definidas a taxa de aprendizagem igual a 100%, a taxa de exploração equivalente a 70% e a taxa de desconto a 90%. A Tabela II apresenta os resultados obtidos após 1000 épocas de treinamento dos agentes.

Tabela II
COMPARAÇÃO ENTRE LABIRINTOS DE DIFERENTES DIMENSÕES

Labirinto	**Tempo (minutos)**	**Confiança Média**	**Pπ**	**PI**
9x9	1,40	78,95%	10	10
15x15	2,21	71,01%	16	16
21x21	6,08	58,23%	40	40
29x29	11,03	54,11%	54	54
49x49	131,36	50,60%	422	201

A partir dos resultados apresentados na Tabela II, pode-se observar que o aumento na dimensão do ambiente provoca um prolongamento no tempo de treinamento. Em 4 de 5 labirintos, o agente foi bem-sucedido em encontrar a saída ótima do labirinto. No entanto, no labirinto de maior dimensão, o agente acabou seguindo um caminho que resultou em mais do que o dobro do número de passos ideais. Isso sugere que o labirinto necessita de mais épocas para a realização da etapa de treinamento. Além disso, nota-se que a confiança média, que o agente apresenta sobre o caminho ótimo aprendido, diminui à medida que a dimensão do ambiente aumenta.

VI. VISUALIZAÇÃO 3D

Adicionalmente às investigações discutidas nas seções anteriores, com o objetivo de aprimorar a compreensão visual durante a exploração de ambientes pelo agente autônomo, uma interface de visualização 3D de ambientes, intitulada como RL *framework*, é também desenvolvida aqui neste trabalho. Nesse contexto, o *software* empregado para o desenvolvimento e implementação da interface RL *framework* foi o Unity. Tal ferramenta, se destaca pela capacidade de oferecer suporte à construção, ambientação e movimentação por meio de *scripts* em C#. Especificamente, para viabilizar a simulação 3D da trajetória percorrida pelo agente, adotou-se um protocolo conhecido como "comunicação via *socket*". Esse protocolo permite a interação entre o *script* Python, responsável pelo treinamento, e o ambiente Unity3D.

A transmissão de dados via *socket* envolve a codificação de um *array* de *bytes* antes do envio e sua subsequente

decodificação para *strings* no destino. No contexto do C#, os dados enviados incluem informações sobre as dimensões e conteúdo da matriz do labirinto, compreendendo as posições de caminho livre e parede. Essa parte do código se concentra na reconstrução do labirinto em C#, permitindo a posterior montagem da visualização 3D.

Após a transmissão dos dados da matriz, um vetor de movimentação é enviado, representando a correlação entre a política resultante e o caminho ótimo para a saída do labirinto. Isso possibilita que o agente se movimente pelo ambiente 3D. A Figura 10 apresenta uma vista panorâmica de um dado labirinto considerado no Unity3D, onde o agente é representado por um carro.

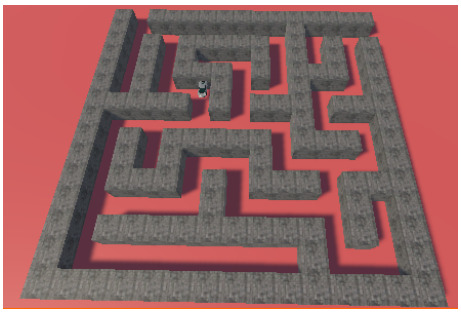

Figura 10. Visualização panorâmica do labirinto 3D

Além da vista panorâmica, a interface RL *framework* também apresenta uma vista em primeira pessoa do agente autônomo. A Figura 11 apresenta a vista em primeira pessoa correspondente ao local em que o agente se encontra na Figura 10. Dessa forma, o desenvolvedor pode visualizar a evolução do treinamento do agente autônomo usando RL. Além disso, na visualização em primeira pessoa, o desenvolvedor pode ter uma percepção da taxa de confiança para cada possível ação a ser realizada no estado atual do labirinto.

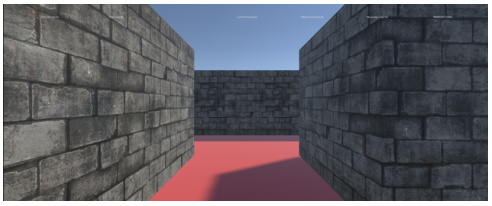

Figura 11. Vista em primeira pessoa do agente no labirinto

VII. Conclusões e Considerações Finais

Neste trabalho de pesquisa, a solução de labirintos usando RL com *Q-learning* foi investigada. Nesse contexto, agentes autônomos foram implementados tendo em vista o aprendizado de políticas ótimas responsáveis pela obtenção das soluções dos labirintos considerados. Foram discutidas estratégias para a obtenção da convergência das soluções, assim como para um mapeamento abrangente do labirinto. Além disso, uma interface de visualização 3D de ambientes, intitulada RL *framework*, foi desenvolvida. Dessa forma, a visualização 3D de labirintos permitiu uma experiência imersiva dos desenvolvedores quanto às dificuldades e incertezas relacionadas às tomadas de decisão em cada estado do labirinto. Finalmente, resultados de simulação numérica foram apresentados e comprovaram a eficácia das implementações dos agentes autônomos, bem como a eficácia do uso do RL *framework* desenvolvido neste trabalho.

VIII. Agradecimentos

Este trabalho foi financiado com recursos do IFSC campus Itajaí.

Referências

[1] I. Watson. (2022, May) Ibm global ai adoption index 2022: New research commissioned by ibm in partnership with morning consult. [Online]. Available: https://www.ibm.com/downloads/cas/GVAGA3JP

[2] R. Sundaram and B. Lubina, "Work-in-progress: Introductory reinforcement learning for student education and curriculum development through engaging mediums," in *Anais Annual Conference of the American Society for Engineering Education (ASEE)*, Jun 26-29 2022.

[3] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, pp. 115 – 152, 1995.

[4] X. Ruan, P. Li, X. Zhu, and P. Liu, "A target-driven visual navigation method based on intrinsic motivation exploration and space topological cognition," *Sci Rep*, vol. 12, no. 3462, 2022.

[5] S. Alamri, S. Alshehri, W. Alshehri, H. Alamri, A. Alaklabi, and T. Alhmiedat, "Robótica autônoma de resolução de labirintos: algoritmos e sistemas," *Journal of Mechanical Engineering and Robotics Research*, vol. 10, no. 12, pp. 668–675, Dec 2021.

[6] S. Alamri, H. Alamri, W. Alshehri, S. Alshehri, A. Alaklabi, and T. Alhmiedat, "An autonomous maze-solving robotic system based on an enhanced wall-follower approach," *IEEE Access*, vol. 11, no. 2, p. 249, Feb. 2023.

[7] P. N. S. Russell, *Inteligência Artificial, 3ª Edição*. Elsevier Brasil, 2014.

[8] B. Gupta and S. Sehgal, "Survey on techniques used in autonomous maze solving robot," in *5th International Conference - Confluence The Next Generation Information Technology Summit*. IEEE, 2014, pp. 323–328.

[9] O. Chang, S. R. Niemes, W. Pijal, A. Armijos, and L. Zhinin-Vera, "Q-learning in a multidimensional maze environment," in *Information and Communication Technologies*, J. Herrera-Tapia, G. Rodriguez-Morales, E. R. Fonseca C., and S. Berrezueta-Guzman, Eds. Cham: Springer International Publishing, 2022, pp. 217–230.

[10] A. Amini, T. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus, "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles," in *Anais Int. Conf. on Robotics and Automation (ICRA)*, May 23-27 2022.

[11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, Mar 2020.

[12] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, Sep 2022.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. London, UK: MIT Press, 2016.

[14] T. Bailey and H. Durrant, "Simultaneous localization and mapping (SLAM) Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, 2006.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. London, UK: MIT press, 2018.

[16] D. Hemkemaier and E. Silva, "Inteligência artificial aplicada em robôs autônomos para a solução de labirintos dinâmicos," in *Proc. of the 8th Workshop of Robotics in Education (WRE)*, Curitiba, PR, 2017, pp. 1–6.

[17] M. P. Méxas, "Um estudo sobre técnicas de busca em grafos e suas aplicações," Master's thesis, Coordenação dos Programas de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 1982.