Localização de Fontes de Gás Usando Aprendizagem por Reforço com Robôs Móveis na Indústria da Mineração

Maurício Souza Sathler [*,†] **Andre Luiz Maciel Cid** [*,†]
Felipe G. Oliveira [****] **Gustavo M. Freitas** [**] **Gustavo Pessin** [*]
Héctor Azpúrua [*,***]

[*] *Instituto Tecnológico Vale (ITV) (e-mail: mauricio.sathler@aluno.itv.org, andre.cid@aluno.itv.org, gustavo.pessin@pq.itv.org, hector.azpurua@itv.org).*
[**] *Departamento de Engenharia Elétrica, UFMG (e-mail: gustavomfreitas@ufmg.br).*
[***] *Departamento de Ciência da Computação, UFMG (e-mail: hector.azpurua@dcc.ufmg.br)*
[****] *Instituto de Ciências Exatas e Tecnologia, UFAM (e-mail: felipeoliveira@ufam.edu.br)*
[†] *Universidade Federal de Ouro preto, UFOP (e-mail: mauricio.sathler@aluno.ufop.edu.br, andre.cid@aluno.ufop.edu.br)*

Abstract: Inspection operations are essential to guarantee operational safety in the mining industry. When carried out in confined environments, these processes demand attention to the various cliffs involved, among them the presence of harmful gases and lack of oxygen. Therefore, this paper presents an implementation of the deep reinforcement learning for gas source tracking using mobile robots. We present the necessary configurations for the simulation of a gas plume, including the parameters and architecture of the deep learning network used for gas source detection. Results generated in representative simulated environments show the technical feasibility of the proposed method and open the way for future applications with real robots.

Resumo: Operações de inspeção são essenciais para garantir a segurança operacional na indústria da mineração. Estes processos, quando realizados em ambientes confinados, demandam atenção pelos diversos riscos envolvidos, entre eles a presença de gases nocivos e falta de oxigênio. Desta forma, o presente trabalho apresenta a implementação do aprendizado profundo por reforço para o rastreio de fonte de gás utilizando robôs móveis. Além das configurações necessárias para a simulação de uma pluma de gás, também são apresentados as configurações, parâmetros e arquitetura da rede de aprendizado profundo utilizada. Resultados gerados em ambientes simulados representativos mostram a viabilidade da técnica proposta e abrem caminho para futuras aplicações com robôs reais.

Keywords: Gas source; Mobile robot; Confined spaces; Reinforcement learning; Simulation;
Palavras-chaves: Fonte de gás; Robótica móvel; Ambientes confinados; Aprendizado por reforço; Simulação;

1. INTRODUÇÃO

A inspeção de ambientes confinados é rotineira em muitas industrias, entretanto, esses ambientes apresentam diversos desafios operacionais que podem colocar em risco a vida dos inspetores. Dentre estes riscos, um dos mais críticos é a presença de gases nocivos à saúde, que possui a maior média de fatalidade por acidente (Burlet-Vienney et al., 2015). Dessa forma, a inspeção e o mapeamento dos gases nocivos presentes nesses ambientes, assim como a localização da fonte de vazamento de gás (se existir) é uma tarefa de suma importância para a segurança operacional.

Considerando as dificuldades mencionadas anteriormente, foi desenvolvido o EspeleoRobô (Cota et al., 2017; Azpurua et al., 2019), que é um dispositivo robótico que busca maior segurança e qualidade para a realização de inspeções em ambientes confinados. Além dos diversos sensores presentes no EspeleoRobô, que permitem o mapeamento em duas e três dimensões, e de câmeras de alta resolução, o robô pode ser equipado com sensores que informam a concentração de diferentes tipos de gases.

Considerando o problema de encontrar a fonte de vazamento de um gás, atualmente as técnicas de rastreamento com a utilização de aprendizado por reforço têm se mostrado promissoras, principalmente pela utilização de técnicas de aprendizado por reforço profundo ou *Deep Reinforcement Learning* (DRL). Com as técnicas de rastre-

amento é possível interpretar o alto grau de complexidade de modelos de dispersão química de gás, denominados como plumas de gás (Niu et al., 2017).

O aprendizado por reforço é dividido em duas partes: (i) agente: é quem irá aprender e tomar as decisões; (ii) ambiente: compreende todo o resto, sendo responsável por interagir com o agente. Esta iteração é contínua, seguindo a lógica, em que o agente ao perceber o estado do ambiente escolhe uma ação e a partir desta ação o ambiente responde gerando um novo estado e a respectiva recompensa da ação tomada. Para o treinamento de agentes com DRL é necessário uma grande quantidade de iterações em comparação com outros métodos de aprendizado, necessitando assim a coleta abrangente de experiências para que o modelo de aprendizado por reforço consiga generalizar o conhecimento aprendido. O processo de experimentação com robôs reais, é frequentemente realizado em ambientes controlados e costuma ser oneroso e demorado. Dessa forma, a utilização de ambientes simulados para treinamentos iniciais dos modelos de DRL é chave para viabilizar o treinamento de situações complexas, sendo necessário apenas o agente (modelo do robô) e o ambiente com o modelo de dispersão química.

Nesse artigo é apresentada uma abordagem para a localização de fontes de gás em ambientes confinados usando robôs móveis terrestres. Para isso, foram considerados dados providos por um sensor de gás simulado, capaz de estimar a concentração de gás em um ponto do ambiente, como entrada para um modelo de aprendizado por reforço profundo. Experimentos em ambientes simulados com um modelo realístico de propagação de gás sem turbulência demostram a viabilidade do método.

2. TRABALHOS RELACIONADOS

A localização de uma fonte de gás implica em encontrar uma posição onde se tem a maior concentração de partículas químicas deste determinado gás. A disposição destas partículas no ambiente sofre influência direta do vento e naturalmente impacta sua difusão. O problema de busca por fontes é estudado amplamente há mais de três décadas (Chen and Huang, 2018). Dentre as principais estratégias que abordam o referido problema destacam-se as técnicas: baseadas em gradiente, bioinspiradas, multi-robô e probabilísticas.

As técnicas puramente baseadas em gradiente (Dhariwal et al., 2004), foram as primeiras a serem implementadas e se baseiam em deslocar o robô no sentido do gradiente de concentração. Essa técnica apresenta bom desempenho em ambientes de distribuição contínua, pois em caso de pequenas descontinuidades na dispersão da pluma esta técnica já perde efetividade. As estratégias bioinspiradas (Davies et al., 2015) têm como fundamento, reproduzir o comportamento de animais que utilizam algum tipo de pluma de odor para busca de alimento ou até para reprodução. As estratégias probabilísticas, que se baseiam em mapas e modelam a localização da fonte do odor como uma distribuição de probabilidade a partir das leituras do sensor de gás.

Tanto as técnicas baseadas em gradiente e bioinspiradas têm vantagem de possuir baixo custo computacional, no entanto, elas são limitadas em relação aos diferentes tipos de plumas e não levam em consideração o ambiente ao redor, sendo difícil a integração com algoritmos de navegação que acabam atrapalhando o próprio algoritmo de busca de fonte de gás. Já as probabilísticas possuem maior custo computacional como desvantagem, e como vantagem é mais fácil a integração com algoritmos de navegação e consegue compreender melhor os diversos tipos de plumas. Nos últimos anos os estudos na área de algoritmos probabilísticos têm ganhado força, principalmente considerando abordagens baseadas em aprendizado de máquina.

Problemas relacionados ao aprendizado por reforço inicialmente foram resolvidos baseados na tabela Q (Q-Table) (Watkins and Dayan, 1992). Esta tabela contém os estados, ações e o valor das ações. O objetivo da tabela é encontrar a politica ótima, ou seja, encontrar as melhores ações a serem executadas em cada estado existente. Poucos trabalhos, baseados em aprendizado por reforço, foram encontrados para resolver o problema de localização e busca de fontes. Niu et al. (2017) e Hu et al. (2019) propuseram estratégias de rastreamento de pluma turbulenta estática para veículos aquáticos baseados em DRL, onde as técnicas propostas são comparadas com estratégias convencionais. Singh et al. (2020) apresentam uma rede baseada em DRL para resolver o problema de rastreamento de fonte de gás para plumas no ambiente aéreo. Adicionalmente, é mostrada a variação do aprendizado, onde é fornecido um vetor de estados anteriores como entrada do modelo de aprendizado, representando a ideia de memória.

3. MÉTODO PROPOSTO

O método proposto para encontrar a fonte de um gás de forma autônoma considera um robô terrestre com arquitetura diferencial, onde sua pose é representada por uma configuração (x, y, α) no espaço 2-D, onde α é a referência de rotação do robô. O robô é capaz de se locomover livremente pelo ambiente, e é equipado com um sensor capaz de medir a concentração de gás em um ponto do mapa.

Este método é desenvolvido para ser treinado de forma continua e paralela em um ambiente virtual preparado com um modelo realístico de dispersão de gás. O modelo completo utilizado pode ser reproduzido e simulado de forma eficiente, maximizando o número de iterações referentes ao treinamento do modelo de aprendizado por reforço. O diagrama de alto nível pode ser observado na Figura 1.

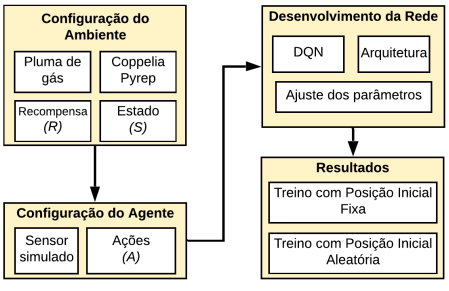


Figura 1. Fluxo de execução das atividades.

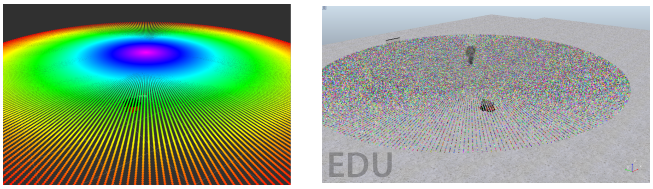
4. CONFIGURAÇÃO DO AMBIENTE

O simulador escolhido para o desenvolvimento da pesquisa foi o CoppeliaSim [1] , sucessor do simulador V-REP. O CoppeliaSim possui um ambiente simulado extensivo, sendo possível representar a cinemática do robô e possibilitando a expansão dos modelos para representação de plumas tridimensionais. Adicionalmente, possui um kit de ferramentas para treinamento de DRL, chamado PyRep [2] .

4.1 Pluma de gás

Uma pluma é uma coluna de fluido em movimento dentro de outro fluido. No caso das plumas de gases nocivos, geralmente as plumas estão se movimento dentro do ar atmosférico, onde esses fluidos podem ser movimentados por difusão ou pressão. As plumas de gás podem ser classificadas em: (i) turbulenta; e (ii) não turbulenta. O número de Reynolds (rey, 2017) é o parâmetro que as classifica, onde em plumas não turbulentas as forças viscosas do meio são dominantes (fluxo lento e baixo número de Reynolds). Estas forças são suficientes para manter todas as partículas alinhadas, resultando em um fluxo laminar, que geralmente ocorre em locais fechados e com pouco fluxo de vento como cavernas e galerias. Já em plumas turbulentas as forças inerciais atuam sobre as forças viscosas (fluxo rápido e alto número de Reynolds), sendo comum em ambientes abertos com grande influência de ventos.

Baseado em modelos de dispersão de plumas odoríficas, foi adaptado para o ambiente simulado o modelo de pluma não turbulenta do tipo radial. Os ambientes confinados como cavernas e tubulações onde atua o EspeleoRobô não possuem fortes correntes de vento que possam gerar turbulências consideráveis na dispersão do gás, sendo a pluma radial o modelo mais adequado para a representação da dispersão nestes ambientes. No modelo radial, a cada incremento na distância do centro da fonte do gás ocorre o decaimento da concentração em todos os sentidos como observado na Figura 2.

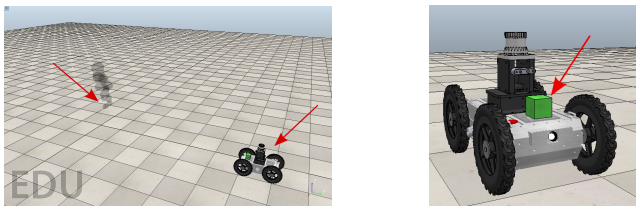


(a) Disposição de concentração em 2D. (b) Pluma turbulenta 2D simulada.

Figura 2. Pluma radial: (a) Disposição de concentração de uma pluma do tipo radial. Os valores centrais na cor roxo possuem a maior concentração de gás e os valores em vermelho nas bordas possuem a menor concentração. (b) Pluma radial 2D implementada ao ambiente do CoppeliaSim.

4.2 Ambiente

O ambiente utilizado foi definido com superfície planar, sem obstáculos, contendo apenas o robô e a pluma de gás (Figura 3a). A plataforma utilizada para a simulação foi o EspeleoRobô (Figura 3b), com configuração de quatro rodas. Neste treinamento outros sensores equipados no robô, como LiDAR e câmeras RGBD, foram desabilitados pois não seriam necessários e com isso é possível melhorar o desempenho. O desempenho é muito importante pois algoritmos de aprendizado por reforço necessitam de um grande número de iterações. Devido a grande quantidade de iterações, pequenos ganhos em tempo são benéficos para reduzir o tempo final de treinamento.



(a) Ambiente com pluma de gás e robô. (b) Plataforma robótica EspeleoRobô.

Figura 3. Ambiente: (a) Ambiente de treinamento preparado no simulador CoppeliaSim contendo fonte de gás e robô móvel. (b) EspeleoRobô e sensor de gás configurado para utilização do PyRep.

Estado Em um ambiente de aprendizado por reforço, a cada passo dado pelo agente, o ambiente deve gerar o próximo estado, a recompensa e verificar se a condição de encerramento do episódio aconteceu. Nesse ambiente os estados são caracterizados pelos seguintes atributos:

- Concentração de gás;
- Distância deslocada entre a posição atual e a posição antes do movimento. Definida na Equação 1, onde D, (x, y) e t consecutivamente representam a distância relativa, posição (x, y) correspondendo ao ponto de referência do simulador e intervalo de tempo;

$$D_t = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \quad (1)$$

- Ângulo α entre a pose atual e a pose posterior ao movimento, podendo variar de $-\pi$ $(-180°)$ a $+\pi$ $(180°)$ (Equação 2);

$$\alpha_t = yaw \quad (2)$$

- Gradiente da concentração de gás onde, caso o gradiente seja positivo, retorna o valor 1 e, caso negativo, retorna o valor 0.

Recompensas No aprendizado por reforço o objetivo do agente é produzido em termos do sinal de recompensa que é recebido do ambiente, pelo fato do agente visar maximizar a soma das recompensas recebidas. Com isso, a função de recompensa deve fornecer um *feedback* positivo em situações onde o robô está dentro da pluma de odor, seguindo o gradiente de concentração e caso encontre a fonte. Caso o robô saia da pluma de odor ou siga um gradiente negativo de concentração de odor, o *feedback* deve ser negativo. Abaixo é mostrado o valor da recompensa recebida pelo agente a cada iteração do episódio $r(s, a)$, onde, r é a recompensa referente a ação a tomada no estado s, e ao fim do episódio todas as recompensas por iteração daquele episódio são somadas e assim é retornada a recompensa final.

[1] `https://www.coppeliarobotics.com/coppeliaSim`
[2] `https://github.com/stepjam/PyRep`

$$r(s,a) = \begin{cases} 1 & \text{Robô dentro da pluma} \\ -1 & \text{Cada iteração do agente com o ambiente} \\ i & \text{Sequência de passos com gradiente positivo} \\ -i & \text{Sequência de passos com gradiente negativo} \\ -100 & \text{10 iterações seguidas fora da pluma} \\ 1000 & \text{Fonte encontrada} \end{cases}$$

5. CONFIGURAÇÃO DO AGENTE

Para definir qual robô corresponde ao agente, qual sua posição e quais juntas devem se movimentar é necessário inicialmente configurar o PyRep.

5.1 Modelagem da plataforma robótica no PyRep

O PyRep fornece uma base inicial de robôs já configurados ao ambiente de aprendizado de máquina como manipuladores, garras e alguns robôs móveis (como o popular Turtlebot). No entanto foi necessário modelar o agente do PyRep para utilizar o modelo do EspeleoRobô (Cid et al., 2020) (Figura 3b), que é o robô real usado para realizar as inspeções. Para configuração da estrutura do modelo de um robô móvel é necessário a configuração correta dos nomes que serão chamados na API PyRep. Dessa forma, foi necessário adaptar o nome e valores das juntas do robô para serem compatíveis, seguindo uma nomenclatura do tipo: *NomeDoRobo*_m_joint*NumeroDaJunta*.

5.2 Ações

Por questão de simplificação, a ação escolhida pelo agente é realizada por 20 passos ou *steps* de simulação, para que assim se atualize o estado do ambiente. É disponibilizado ao agente 5 ações:

$$\begin{matrix}\text{Ações} \\ (20\ steps)\end{matrix} = \begin{cases} 0 & \text{Andar em linha reta} \\ 1 & \text{Parábola a direita} \\ 2 & \text{Parábola a esquerda} \\ 3 & \text{Giro no sentido horário no próprio eixo} \\ 4 & \text{Giro no sentido anti horário no próprio eixo} \end{cases}$$

5.3 Sensor de gás simulado

O sensor de gás é representado pela caixa verde acoplada ao robô, conforme a Figura 3b. Para coleta da concentração de gás pelo sensor, foi desenvolvido um algoritmo de busca da seguinte maneira: Na pluma de odor radial, os pontos distribuídos no ambiente possuem listas de posição (X, Y, Z) e concentração $(\mathbb{P})$, relativos a um ponto fixo (objeto *Dummy*) denominado como "*world*".

Para saber se as partículas de gás se encontram dentro dos limites do sensor, primeiramente é utilizado um método que busca na lista com informações da posição (X, Y, Z) e a intensidade de cada partícula. Vale ressaltar que o sensor simulado tem formato cúbico com lados de 8 centímetros. Em seguida, as partículas são analisadas de forma que os pontos lidos pelo sensor são os que satisfazem as condições seguintes:

$$\phi_x(t) - \frac{T}{2} <= \psi_x(t) <= \phi_x(t) + \frac{T}{2}, \tag{3}$$

$$\phi_y(t) - \frac{T}{2} <= \psi_y(t) <= \phi_y(t) + \frac{T}{2}, \tag{4}$$

onde $\psi_{\{x|y\}}$ é a posição global da partícula química, $\phi_{\{x|y\}}$ é a posição global do sensor de gás no eixo x ou y no instante t, e T o tamanho de um lado do sensor.

Para o calculo de concentração $\mathbb{C}$ das partículas $\mathbb{P}$ que se encontram dentro do sensor foi utilizada a formula:

$$\mathbb{C} = \frac{\sum_{n=1}^{p} \mathbb{P}_n}{|\mathbb{P}|}, \tag{5}$$

onde $|\mathbb{P}|$ é o número total de partículas dentro do sensor.

6. DESENVOLVIMENTO DA REDE

Tanto o Deep Q Learning quanto o Q Learning possuem como entrada o estado s e saída a ação a, porém, o que os difere são como os estados são tratados após a entrada. O Q Learning é um método baseado na tabela Q. Nestes métodos o agente utiliza uma tabela/matriz que contém todos os estados e ações possíveis para tomar suas decisões. Já no Deep Q Learning, método baseado em rede neural, a tabela é substituída por uma rede neural que faz a aproximação da função de valor Q das ações possíveis do presente estado, gerando outros desafios.

Métodos de aprendizado por reforço tradicionais como *Q-Learning* ainda são bastante utilizados, porém, há uma limitação em sua utilização. Essa limitação é percebida quando se trabalha com ambientes grandes ou contínuos, onde para representar esses ambientes seria necessária uma tabela extremamente grande. Tabelas Q muito grandes apresentam baixa eficiência, podendo representar o ambiente de forma não adequada. Uma alternativa é utilizar redes neurais para substituir a Tabela Q e conseguir reproduzir uma quantidade muito maior de estados e ações.

Para a resolução deste problema, foi escolhido utilizar uma arquitetura de *Deep Q Network* (DQN). Por se tratar de um ambiente contínuo, o número de estados e ações se torna muito grande, sendo extremamente custoso representá-lo com a *Q-table* tradicional (Mnih et al., 2015). Por exemplo, no caso da detecção de gás com o ambiente e ações definidos neste trabalho, a tabela Q tradicional teria que ser de um tamanho de aproximadamente 5×10^{12} valores, o que é computacionalmente inviável.

6.1 Deep Q Network

É comprovada a instabilidade do aprendizado por reforço quando se utiliza redes neurais como uma função de aproximação (Thrun and Schwartz, 1993). Isto se deve ao fato da superestimação sistemática dos valores e do *Q-learning* aprender com cada uma das tuplas de experiência Estado, Ação, Recompensa, Estado Seguinte (S, A, R, S') em ordem sequencial, podendo ser influenciado pelos efeitos de correlação entre estes componentes. Porém, é possível estabilizar o treinamento com duas técnicas:

- *Experience Replay:* Possibilita a reutilização de experiências anteriores no aprendizado. Sem essa técnica,

as experiências são descartadas após seu uso. Com *Experience Replay* isso não acontece. Essa técnica armazena as experiências passadas em um buffer e as misturam dentro deste, quebrando a correlação entre experiências e melhorando dois casos do aprendizado por reforço: (i) estabilização do aprendizado; e (ii) reutilização de experiências passadas;

- *Target network:* Quando utilizadas as redes neurais para representar a função de valor, diferente do *Q-learning* que atualiza o exato valor de estado, no DQN se ajusta os parâmetros de toda a rede que podem afetar o valor de ação do próximo estado. A *target network* (Figura 4) propõe resolver esse problema com outra DQN de mesma configuração. Essa rede *target* será atualizada com uma frequência menor com o objetivo de fixar os valores de ação dos próximos estados por um determinado período de tempo, estabilizando o aprendizado.

  Existem dois métodos de atualização da rede. O *full update*, que atualiza todos parâmetros da rede, e o *soft update*, em que a rede de destino é atualizada com maior frequência, mas em pequenas partes e em escala menor. Assim, a rede *target* irá se mover lentamente e sua atualização deve ser frequente para que o efeito seja perceptível. O valor de τ é usado na formula de *soft update* como mostra a equação 6, onde θ são os parâmetros da rede destino, θ^- os parâmetros da rede *target* e τ é a constante de atualização:

$$\theta^- = (\theta * \tau) + (\theta^- * (1 - \tau)). \tag{6}$$

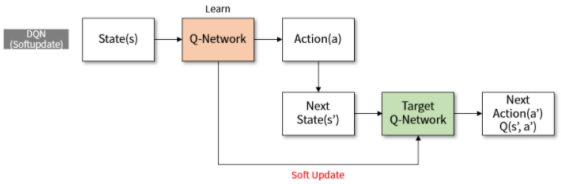


Figura 4. Arquitetura de uma Deep Q-Network (DQN).

6.2 Função de Custo

No aprendizado de máquina, a função de perda é determinada como a diferença entre a saída real e a saída prevista do modelo para o exemplo de treinamento único, enquanto a média da função de perda para todo o exemplo de treinamento é denominada como a função de custo. Esta diferença calculada das funções de perda é denominada como o valor de erro. No caso, foi utilizado como função de custo o erro quadrático médio (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2, \tag{7}$$

onde n é o número total de dados, y_i o valor real e y_i' o valor previsto.

6.3 Normalização

A normalização é importante para igualar o peso de cada dado de entrada na rede. Isto pois dados com faixas de valores muito diferentes dificultam os otimizadores e geram oscilações no aprendizado, resultando em acréscimo de tempo para convergência. Assim, foi adotado o padrão de normalização, que é compreender todos os dados entre 0 e 1. As entradas da rede foram normalizadas da seguinte forma:

- No caso da pluma radial, a leitura do sensor (Equação 5) pode variar de 0 a 400. Logo, o valor normalizado (L_N) é encontrado dividindo o valor de leitura (L) por 400:

$$L_N = \frac{L}{400} \tag{8}$$

- A maior distância percorrida pelo robô em um único movimento tem valor próximo de 2 metros. Então, a distância caminhada é dividida por 2:

$$D_N = \frac{\sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}}{2} \tag{9}$$

- O ângulo da pose do robô está em radianos e compreende de $-\pi$ a $+\pi$, logo, o ângulo normalizado (α_N) é encontrado adicionando π e dividindo por 2π:

$$\alpha_N = \frac{\alpha_t + \pi}{2 * \pi} \tag{10}$$

6.4 Arquitetura

A arquitetura da rede é onde se define quais serão as entradas, camadas ocultas, funções de ativação e saídas da rede. Baseado no trabalho de Singh et al. (2020), os estados passados foram salvos em vetores no intuito de reproduzir a ideia de memória em algumas entradas, conforme mostra a Figura 5. Foram testadas algumas arquiteturas e parâmetros e a que obteve melhores resultados foi a que possui 16 entradas, 2 camadas ocultas com 64 neurônios em cada, funções de ativação tangente hiperbólica e 5 ações de saída.

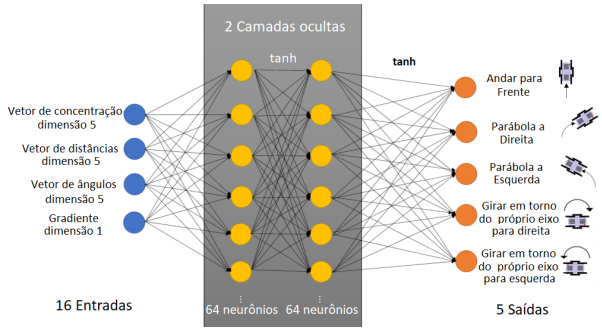


Figura 5. Arquitetura utilizada.

6.5 Parâmetros

Além da rede neural profunda também é necessário o ajuste dos parâmetros de aprendizagem para que este aprendizado ocorra da melhor maneira possível. Os parâmetros utilizados nos treinamentos foram:

- Buffer Size: 500.000, este é responsável pelo armazenamento de experiências e teve seu valor definido com intuito de armazenar todas elas. No experimento com posição inicial fixa, cada episódio tem 50 experiências. Multiplicando este valor pelo número de épocas máximo configurado (10.000), gera o total de 500.000 experiências;

- Batch Size: 3.200, representa o tamanho do lote de experiências que serão utilizadas para o aprendizado;
- Gamma: 0,95, é o fator de desconto;
- Tau: 0,001, é o parâmetro de interpolação da função *soft update* para atualizar os parâmetros da rede target;
- Learning Rate: 0,0003, é a taxa de aprendizado;
- Update_every: 1, é a taxa de atualização, onde a rede target é atualizada a cada nova experiência;
- Otimizador: Adam.

7. RESULTADOS

Os testes foram executados em um computador HP 8GB de memória RAM, processador Intel i5-5200U CPU, com sistema operacional Ubuntu 16.04. Como mencionado na Seção 4.1, o ambiente virtualizado foi desenvolvido no CoppeliaSim V4.0.0.

7.1 Treinamento com posição inicial fixa

Para o treinamento com posição inicial fixa, foi utilizada a configuração apresentada na Figura 5 e os parâmetros definidos na seção 6.5, com um épsilon inicial igual a 1 e decaimento de 0,99 por episódio completo. O tamanho do episódio foi definido com 50 iterações, podendo ser finalizado antes caso ocorra alguma das duas condições de parada: (i) leitura do sensor de gás maior que 350 (fonte localizada); ou (ii) mais de 10 iterações seguidas com leitura do sensor igual a 0 (perda da pluma). O resultado do treinamento pode ser observado na Figura 6 onde, por volta de 1200 episódios, foi obtida uma porcentagem de acerto superior a 99%. No entanto, apesar de alcançar uma alta porcentagem de acerto, nesse aprendizado não é possível generalizar o conhecimento, ele apenas é efetivo quando o robô é posicionado na posição especifica. Dessa forma, é necessário realizar um novo treinamento, considerando posições iniciais aleatórias.

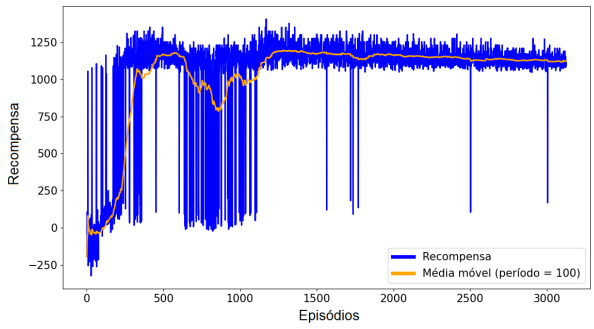


Figura 6. Gráfico de recompensa por episódio em treinamento com posição inicial fixa.

7.2 Treinamento com posição inicial aleatória

Para realização do experimento com posição inicial aleatória o robô deve ser disposto em uma posição qualquer dentro da pluma de gás, e regiões onde o robô aparece bem próximo à fonte são descartadas. A configuração da rede e parâmetros se manteve a mesma da seção 6.5, as modificações foram: o número de iteração por episódio deve ser 50 nos primeiros 650 episódios, após isso, é aumentado para 150 iterações por episódio. Isto o robô pode iniciar em alguma posição que 50 iterações não é suficiente para chegar à fonte. É importante ressaltar que muitas iterações nos episódios iniciais atrasam o tempo de treinamento.

O resultado do treinamento é ilustrado na Figura 7. Após o treinamento, com intuito de validação é carregado o agente treinado por 300 episódios onde em 297 o agente consegue encontrar a fonte de gás, isto significa uma taxa acerto de 99%. Finalmente, na Figura 8, demonstra duas instâncias de localização de fonte de gás, visualizadas no ambiente simulado, onde os caminhos percorridos pelo robô (odometria) são sinalizados pelas linhas pretas e os diferentes pontos de partida sinalizados por círculos brancos. Ao fim do experimento e da validação é possível notar que a rede conseguiu generalizar o conhecimento e encontrar a fonte de gás mesmo iniciando em posições não presentes no treinamento. Vale ressaltar que o robô com os movimentos circulares começou, de certo modo, a entender para onde o vetor do gradiente da concentração do gás está apontando. Porém, um caminho considerando apenas o gradiente de concentração é difícil de ser gerado, isso ocorre pois só ficam disponíveis cinco leituras da concentração de gás e posição relativa do robô. Outro fator que influência é o fato da simplificação das ações do robô, onde cada movimento é executado por 20 *steps*. Como visto na Figura 8 (b), por não utilizar muitos pontos, o robô acaba executando o movimento duas vezes.

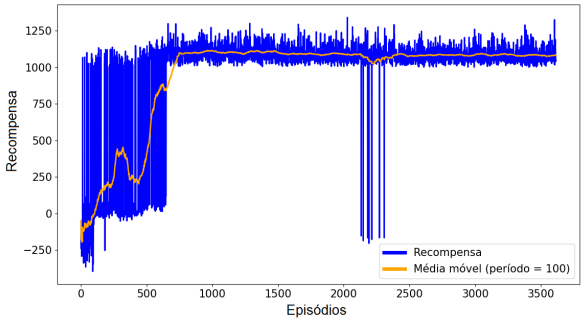


Figura 7. Resultado do treinamento com posição inicial aleatória.

8. CONCLUSÃO

Neste artigo foi apresentada uma aplicação do algoritmo de aprendizado profundo por reforço DQN para rastreamento de fontes de gás em robôs móveis terrestres. Resultados iniciais em ambientes realísticos com plumas de gás radial mostraram que o algoritmo de aprendizado por reforço consegue generalizar o conhecimento para encontrar a fonte da pluma de gás mesmo com o robô partindo de posições aleatórias do mapa, onde ele nunca foi especificamente treinado. O desenvolvimento do ambiente de simulação e as otimizações realizadas na implementação foram chave para permitir que o grande número de iterações no treinamento fosse viável.

Os resultados deste trabalho propiciam a investigação de várias linhas pesquisa: melhores métodos de treinamento de aprendizado por reforço e diferentes tipos de modelagem, recompensas e variações nas redes DQN. Trabalhos futuros vão focar na busca de fontes de gás em ambientes com obstáculos, considerando os dados de sensores como

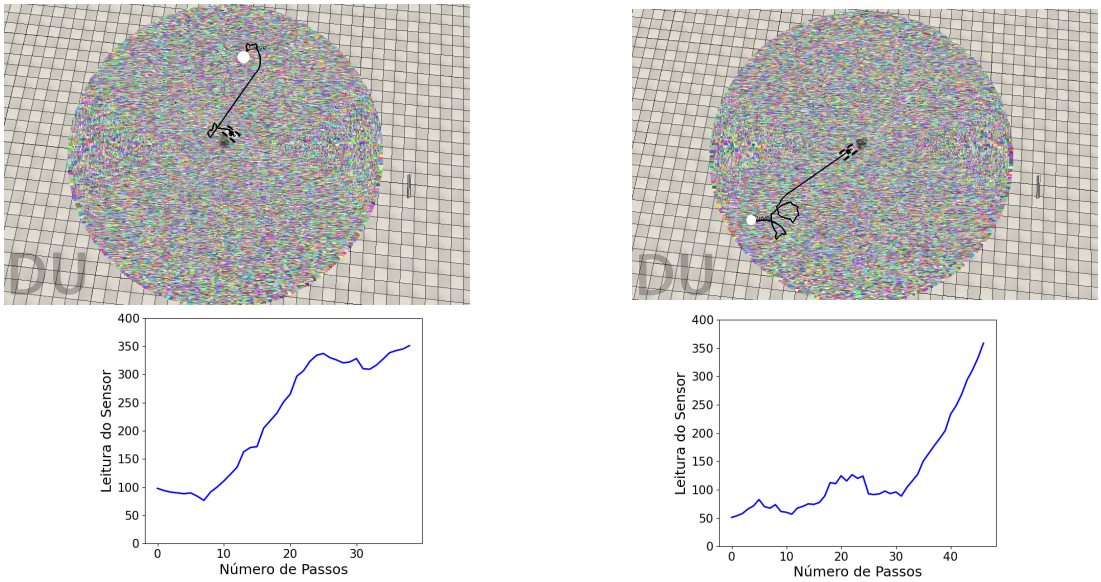


Figura 8. Experimento com posição inicial do robô randomizada (duas repetições). Na fila superior podem ser visualizados a posição inicial do robô (circulo branco) e o caminho percorrido (linha preta). Na fila inferior pode ser observado a concentração de gás ao longo do tempo para cada experimento.

LiDAR nas recompensas. Também serão considerados outros tipos de plumas, como plumas turbulentas em 2D e 3D, assim como o uso de outras arquiteturas de robôs, como plataformas aéreas não tripuladas (VANTs).

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Código de Financiamento 001; do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); do Instituto Tecnológico Vale (ITV); da Universidade Federal de Ouro Preto (UFOP); da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG); e da Vale S.A.

REFERÊNCIAS

(2017). Bibliography. In B.E. Rapp (ed.), *Microfluidics: Modelling, Mechanics and Mathematics*, Micro and Nano Technologies, 745–754. Elsevier, Oxford.

Azpurua, H., Rocha, F., Garcia, G., Santos, A.S., Cota, E., Barros, L.G., Thiago, A.S., Pessin, G., and Freitas, G.M. (2019). EspeleoRobô - a robotic device to inspect confined environments. In *2019 19th Int. Conf. on Advanced Robotics (ICAR)*. IEEE.

Burlet-Vienney, D., Chinniah, Y., Bahloul, A., and Roberge, B. (2015). Occupational safety during interventions in confined spaces. *Safety Science*, 79, 19–28.

Chen, X. and Huang, J. (2018). Odor source localization algorithms on mobile robots: A review and future outlook. *Robotics and Autonomous Systems*, 112.

Cid, A., Nazário, M., Sathler, M., Martins, F., Domingues, J., Delunardo, M., Alves, P., Teotônio, R., Barros, L.G., Rezende, A., Miranda, V., Freitas, G., Pessin, G., and Azpúrua, H. (2020). A simulated environment for the development and validation of an inspection robot for confined spaces. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 1–6.

Cota, E., Torre, M.P., Rocha, F.A.S., Garcia, G., Ângelo Junior, Ramos, V., Queiroz, V., Zanini, V., Brito, G., Marques, A., Érica Pinto, Nogueira, L., Freitas, G., Miola, W., dos Reis, M.A., Araújo, R., and Brandi, I. (2017). Dispositivo de monitoramento remoto de cavidades-espeleorobô. *SBAI - Simpósio Brasileiro de Automação Inteligente*, 1761–1762.

Davies, A., Louis, M., and Webb, B. (2015). A model of drosophila larva chemotaxis. *PLOS Computational Biology*, 11, e1004606.

Dhariwal, A., Sukhatme, G., and Requicha, A. (2004). Bacterium-inspired robots for environmental monitoring. In *IEEE ICRA 2004*, volume 2, 1436–1443 Vol.2.

Hu, H., Song, S., and Chen, C.L.P. (2019). Plume tracing via model-free reinforcement learning method. *IEEE Transactions on Neural Networks and Learning Systems*, 30(8), 2515–2527.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Niu, L., Song, S., and You, K. (2017). A plume-tracing strategy via continuous state-action reinforcement learning. In *2017 Chinese Automation Congress (CAC)*.

Singh, S., van Breugel, F., Rao, R., and Brunton, B. (2020). Understanding biological plume tracking behavior using deep reinforcement-learning. 750–752.

Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Connectionist Models Summer School*.

Watkins, C.J.C.H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279–292. URL `http://jmvidal.cse.sc.edu/library/watkins92a.pdf`.