Manobra de exploração e procura em ambiente desconhecido

António Carneiro
Mestrado de Sistemas Autónomos
Instituto Superior de Engenharia do Porto
Porto, Portugal
1171085@isep.ipp.pt

João Pedro Oliveira
Mestrado de Sistemas Autónomos
Instituto Superior de Engenharia do Porto
Porto, Portugal
1170547@isep.ipp.pt

***Abstract*—Development of a control maneuver for a mobile robot on a simulated environment that allows the search for an objective in unknown surroundings**

***Index Terms*—Python, ROS, Linux, project DISRUPTIVE**

I. Introdução

O trabalho realizado enquadra-se no conteúdo pedagógico da Unidade Curricular de Controlo de Sistemas Autónomos (CSIAU), do primeiro ano do Mestrado em Sistemas Autónomos do Instituto Superior de Engenharia do Porto.
Para a realização deste trabalho foi usado um ambiente de simulação *Open Source*, fornecido durante a realização da prova **Hackathon em ROS - Um desafio na Robótica Móvel**, organizada pelo *Research Centre in Digitalization and Intelligent Robotics* (CeDRI), no âmbito do projeto DISRUPTIVE, que decorreu de 14 a 18 de junho de 2021.
Este artigo centra-se no desenvolvimento de um manobra que controlo, executada por robô móvel presente na simulação, que permita que este seja capaz de procurar fontes de gás (geradas aleatóriamente no mundo), de forma autónoma e num período de tempo útil, sem conhecimento prévio do ambiente. A simulação e o robô móvel nela presente serão descritos com maior detalhe no cap.III

II. Hackathon em ROS - Um desafio na Robótica Móvel

Este *Hackathon* foi desenvolvido tendo como público alvo estudantes de engenharia eletrotécnica, computação, mecânica e entusiastas da robótica. Tinha como objetivos o desenvolvimento de código em *Python* para o controlo de um robô num ambiente desconhecido e como requisitos para participação o conhecimento de *Python* e ROS e a familiarização com conceitos de geometria, álgebra e robótica móvel.
Após a divulgação feita pelo ISEP, foi feita a inscrição de uma equipa composta por 3 estudantes do primeiro ano de mestrado (António Carneiro, Carlos Silva e João Oliveira), onde foram completadas todas as provas, não conseguindo, no entanto, vencer o desafio final.

III. Ambiente de Simulação

O ambiente de simulação usado é constituído por três componentes principais, um mapa 2D, um tópico ROS criado para simular uma ”fonte” de gás e um robô móvel representado por um quadrado vermelho como poderá ser observado na Fig.1.

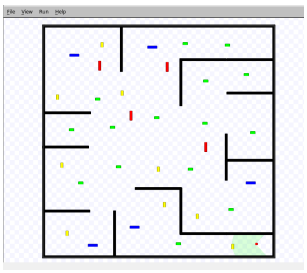

Fig. 1. Mapa original

Esta simulação foi escrita em *Python* e usa ROS para interligar todos os seus componentes. Nos próximos capítulos será feita uma descrição de cada um dos componentes principais.

A. Mapa 2D

O mapa é constituido por paredes (representadas a preto) e blocos (representados a cores) pelos quais o robô não consegue atravessar. Caso o robô entre em contacto com estes elementos a simulação para automáticamente.
Existem 5 mapas diferentes, que são sorteados aleatóriamente aquando do início da simulação, alterando o perfil das paredes e a posição dos blocos como podemos observar na Fig.2

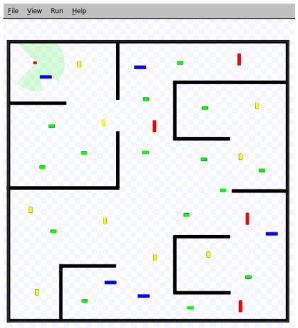

Fig. 2. Mapa Gerado Aleatóriamente

B. Robô Móvel

O robô móvel é exibido como um quadrado vermelho por motivos de simplicidade. Este segue um modelo de tração diferencial, representado na Fig.3 e está equipado com um sensor LIDAR com um FOV (*Field of view*) de 270 graus (centrado com a orientação do robô) um GPS e um sensor de gás, simulado pela subscrição de um tópico como que será discutido no sub-capítuloIII-C. Este robô é guiado alterando os valores de velocidade linear e velocidade angular e publicando no tópico "/cmd_vel".

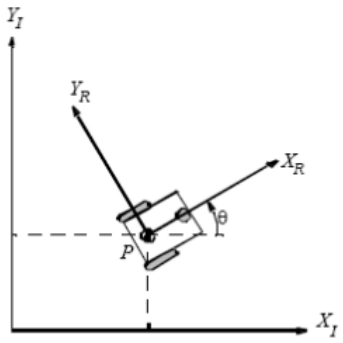

Fig. 3. Esquema Robô Tração diferencial

O LIDAR usado na simulação, é uma versão virtual de um *LMS200*. Possui um alcance entre 0 e 5 metros e publica no tópico "/base_scan" um *timestamp* e dois vetores, com 270 colunas cada, onde cada coluna corresponde a 1 grau no FOV. Estes vetores representam os valores de *Range* e *Intensity*, onde cada valor de *Range* varia entre 0 e 5 metros e cada valor de *Intensity* varia entre 0 e 1, tomando o valor 0 quando para aquele ângulo não tiver sido encontrado nenhum obstáculo dentro dos valores de *Range* e 1 quando tiver sido encontrado um obstáculo dentro dos valores de *Range*.

C. Geração do Gás

O sensor de gás é simulado através da subscrição do tópico "/Gas". Neste tópico, é publicado um valor percentual entre 0 e 100, que relaciona a posição atual do robô com a posição da fonte de gás no mundo. Este parâmetro está contaminado por um ruído gaussinade Quanto mais alto for este valor, mais próximo está o robô da fonte de gás. O cálculo deste parâmetro é feito da seguinte forma,

$$Gas = 100 \times e^{-\frac{d^2}{100}} \quad (1)$$

Sendo d a distância euclidiana entre a posição da fonte de gás no mundo e a posição do robô.
A partir do momento em que o robô se começa a aproximar da fonte de gás (na simulação usada foi escolhido o valor de 23 m), a variável "Gas" é contaminada com ruído gaussiano de $\mu = 0$ e $\sigma = 0.5$ de modo a dificultar a localização.

IV. Paradigmas de Controlo

Um paradigma, de um ponto vista contextual, corresponde a uma filosofia ou conjunto de regras e/ou técnicas que caracterizam a abordagem a uma classe de problemas. No decorrer da disciplina abordamos os três grandes paradigmas clássicos de controlo, o deliberativo, o reativo e o híbrido. O paradigma deliberativo é caracterizado por uma filosofia de *"Think hard, act later"* onde o planemanto e o raciocínio são impostos para a resolução de problemas, já no paradigma reativo impõe-se uma filosofia oposta, *"Don't think, react!"*, onde a "inteligência" advém da organização de comportamentos reativos simples [1]. O paradigma híbrido combina o melhor dos dois mundos, aplicando uma componente deliberativa para o planemanto estratégico e outra reativa para os comportamentos mais básicos e de mais baixo nível.
Para este trabalho optamos por usar uma arquitetura de controlo reativa, criando um conjunto de 4 comportamentos simples, descritos no cap.IV-B que quando aplicados em simultâneo, regidos por um modelo de arbitragem, descrito no cap. IV-C conseguem completar a tarefa de encontrar a fonte de gás no mundo.

A. Arquitetura de Controlo

Como já foi descrito anteriormente, a arquitetura de controlo implementada neste artigo, baseia-se no paradigma reativo, onde o aparecimento de comportamentos complexos como navegar num ambiente desconhecido e encontrar uma fonte de gás, emergem da junção de vários comportamentos mais simples. O esquema da Fig.4 ilustra a interação destes diferentes comportamentos,

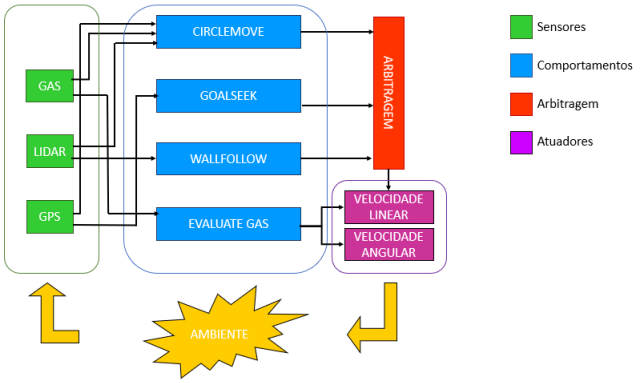

Fig. 4. Esquema da arquitetura de controlo implementada

B. Comportamentos

A abordagem escolhida para resolver o problema de encontrar gás em tempo útil foi a implementação de 4 comportamentos, representados a azul na Fig.4. Essencialmente, o *GOALSEEK* permite ao robô vaguear pelo mundo, enquanto o *WALLFOLLOW* impede que este choque com obstáculos e o *EVALUATE GAS* verifica a concentração de gás no local onde o robô se econtra. Por fim, teremos que abordar o ponto principal do problema. Visto que não sabemos onde se localiza

a fonte do gás, teremos que explorar o mapa escolhendo uma direção que apresente uma maior concentração de gás. Para isto foi implementado o *CIRCLEMOVE*, que tal como o nome indica, obriga o robô a realizar um pequeno percurso circular e através dos valores de gás lidos nesse percurso, obtêm uma nova direção.

1) GOALSEEK: O modo de operação *GOALSEEK* permite essencialmente ao robô explorar o mapa. Este comportamento atribui uma direção e uma distância que o robô deverá percorrer. Inicialmente a direção é dada como "andar" em frente, mas a partir do momento que é detetado gás, esta direção começa a ser refinada e pesada com a direção dada pelo comportamento *CIRCLEMOVE*, o que permite ao robô deslocar-se no sentido de maior concentração de gás.

No modo GOALSEEK, a velocidade angular aplicada é a resultante de um peso entre a velocidade angular da iteração anterior e do ângulo relativo entre a orientação do robô face ao mundo e o ângulo calculado entre a sua posição atual e a direção desejada. De modo a tentar suavizar a rotação, estas velocidades angulares são pesadas, de acordo com a Eq.2.

$$angular.z = 0.6 \cdot \omega_{t-1} + 0.4 \cdot \ \omega_t \tag{2}$$

2) WALLFOLLOW: O modo *WALLFOLLOW* é invocado sempre que é encontrado um obstáculo no caminho do robô. O objetivo principal é encontrar uma nova direção, que o robô deverá seguir de modo a não haver contacto com os obstáculos. Para tal é realizada uma análise dos valores lidos pelo sensor LIDAR, sendo escolhida a direção não obstruída mais próxima da direção préviamente desejada.

3) EVALUATE GAS: O comportamento *EVALUATE GAS* é um comportamento bastante simples, que testa continuamente se a concentração de gás está próxima do valor máximo do sensor (100%). Se estiver, o comportamento pára o robô e a tarefa de procurar gás e declarada como concluída.

4) CIRCLEMOVE: Após ser percorrida a distância determinada pelo comportamento *GOALSEEK*, e ainda não ter sido encontrada a fonte do gás, será procurada uma nova direção a seguir. Para isto foi implementado o modo *CIRCLEMOVE*, onde o robô realiza um pequeno percurso circular e lê a informação relativa ao gás durante o processo.

Após completar o movimento circular, a nova direção é dada pelo vetor que passa do centro do círculo até ao ponto com maior valor de gás registado no movimento circular. Para calcular o centro do círculo, são escolhidos 3 pontos no percurso (P1, P2 e P3), como podemos observar na Fig.5, sendo definidas duas retas a partir destes pontos.

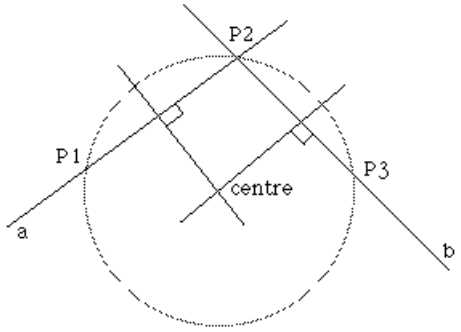

Fig. 5. Determinar centro de uma circunferência a partir de 3 Pontos [2]

A interseção das mediatrizes das retas será o centro da circunferência, tal como podemos observar na Eq.7 .

$$y_a = m_a(x - x_1) + y_1 \tag{3}$$

$$y_b = m_b(x - x_2) + y_2 \tag{4}$$

Com os declives dados por:

$$m_a = \frac{y_2 - y_1}{x_2 - x_1} \tag{5}$$

$$m_b = \frac{y_3 - y_2}{x_3 - x_2} \tag{6}$$

A interseção final:

$$x = \frac{m_a m_b(y_1 - y_3) + m_b(x_1 + x_2) - m_a(x_2 + x_3)}{2(m_b - m_a)} \tag{7}$$

O valor de y pode ser obtido substituindo x numa das Eq 3 ou 4.

C. Arbitragem

Para a arbitragem dos diversos comportamentos, foi escolhido o método de arbitragem de prioridade fixa. Neste método, os comportamentos de prioridade mais elevada serão sempre executados desde que esteja a decorrer uma tarefa de menor prioridade.

Na implementação detalhada neste artigo, os comportamentos tomam a seguinte ordem decrescente de prioridade, *EVALUATE GAS* (caso seja ativado "desliga" o robô), seguido do *WALLFOLLOW*, *GOALSEEK* e *CIRCLEMOVE*.

V. Análise de Resultados

Executando a simulação para o mapa da Fig.1, poderemos analisar o desempenho do algoritmo no cumprimento da tarefa de encontrar gás.
Analisando a trajetória do robô ao longo das várias iterações (Fig.6), podemos observar que este executa o comportamento

CIRCLEMOVE duas vezes, não vagueando de forma exagerada pelo mapa, demonstrando assim que o algoritmo é eficaz a resolver o problema.

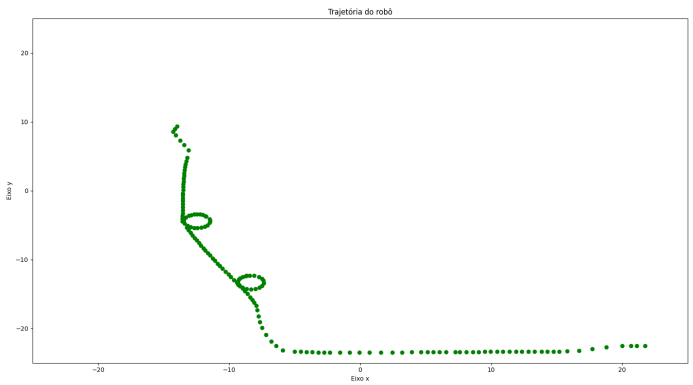

Fig. 6. Trajetória do robô

Um melhor meio de comparação, seria observar a intensidade de gás ao longo da trajetória. Para isto, foi usada a Eq.1, de modo a podermos construir um *colormap* onde as cores mais claras representam uma maior intensidade de gás e as cores mais escuras representam uma menor intensidade de gás.

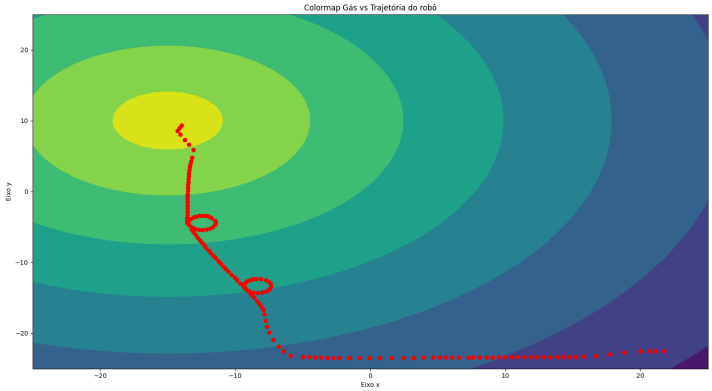

Fig. 7. *Colormap* do gás vs Trajetória do robô

É possível observar que o robô vai conseguindo evitar os obstáculos e direcionar-se para zonas com crescente quantidade de gás até encontrar a fonte.

Verificando agora o valor da distâmcia normal do robô á fonte de gás, Fig.8, constatamos novamente que este valor apresenta no geral uma tendência decrescente ao longo das itereações, validando assim o algoritmo.

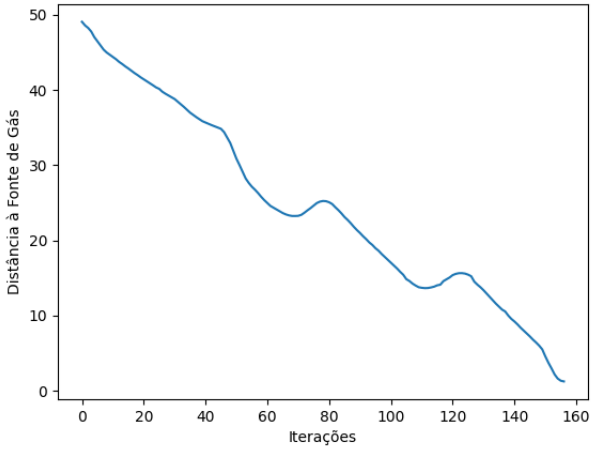

Fig. 8. Distância normal desde a fonte de gás ao robô ao longo das iterações

VI. Conclusão

Este foi sem dúvida um trabalho interessante, onde pudemos aplicar conhecimentos adquiridos ao longo da discilplina e fortalecer as nossas capacidades de resolução de problemas, programação em *python* e ROS. Consideramos que o objetivo deste artigo foi cumprido, o robô consegui alcançar a fonte de gás apenas sabendo a sua localização no mundo e uma leitura sensorial do valor de gás. Cumprindo assim, também, os objetivos da disciplina para este trabalho, implementando diferentes manobras de controlo que continham *feedback*.

References

[1] Alfredo Martins, “Arquiteturas de Controlo”, CSIAU (2021), Acessado: 04 Set. 2021. [Online]. Disponível: https://moodle.isep.ipp.pt/pluginfile.php/135512/mod_resource/content/1/csaut21_arqs.pdf

[2] Paul Bourke, ”Circles and spheres”, 1992, Acessado: 29 Ago. 2021. [Online]. Dispon]ivel: http://paulbourke.net/geometry/circlesphere/