# Exploração Autônoma de Ambientes Planares Utilizando um Dispositivo Robótico Móvel\*

Lucca Leão \* Israel Amaral \* Héctor Azpúrua \*\*,\*\*\* Adriano Rezende \*
Gustavo Pessin \*\*\* Gustavo Freitas \*

\* Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais

\*\* Dept. Ciência da Computação, Universidade Federal de Minas Gerais

\*\*\* Instituto Tecnológico Vale (luccaleao98, israelfilipe, adrianomcr, gustavomfreitas)@ufmg.br (hector.azpurua, gustavo.pessin)@itv.org

Abstract: This paper presents a system for autonomous exploration of planar environments using mobile robots, developed focusing on the implementation in the EspeleoRobô, a robotic device capable of inspecting confined environments. The exploration is carried out based on the detection of frontiers located at the edge of the environment known by the robot. When moving towards the identified frontiers, the robot generates a map of the traversed environment until the entire operating site is explored. For that, we employed a simultaneous localization and mapping strategy using a particle filter. During exploration, the identified frontier cells are clustered based on the distance between them. We propose two simple strategies to explore the environment, prioritizing navigation to the nearest frontier or the frontier with the highest number of cells. The robot movement is performed using a Dijkstra algorithm for path planning and a navigation control by artificial vector fields. The exploration system is validated through simulations and experiments with a real robot, creating complete maps representing the covered environments. The obtained results indicate a better performance while exploring the nearest frontiers, with smaller distances covered and fewer actions performed by the robot.

Resumo: Este artigo apresenta um sistema de exploração autônoma de ambientes planares utilizando robôs móveis, desenvolvido com foco na implementação no EspeleoRôbô, um dispositivo robótico para a inspeção de ambientes confinados. A exploração é realizada com base na detecção de fronteiras localizadas no limite do ambiente conhecido pelo robô. Ao se mover em direção às fronteiras identificadas, o robô gera um mapa do ambiente percorrido, até que todo o local de operação seja explorado. Para tal, uma estratégia de localização e mapeamento simultâneos por filtro de partículas é empregada. Durante a exploração, as células de fronteira são agrupadas com base na distância entre elas. A exploração do ambiente pode ser realizada utilizando duas estratégias simples, priorizando a navegação até a fronteira mais próxima ou até a fronteira com maior número de células. A movimentação do robô é realizada utilizando um algoritmo de Dijkstra para planejamento de caminhos e um controle de navegação por campos vetoriais artificiais. O sistema de exploração é validado em simulações e experimentos com um robô real, sendo capaz de criar mapas completos dos ambientes percorridos de maneira satisfatória. Os resultados obtidos indicam um melhor desempenho da estratégia de exploração das fronteiras mais próximas, com menores distâncias percorridas e um menor número de ações realizadas pelo robô.

Keywords: Mobile Robotics; Frontier Exploration; Localization and Mapping; Path Planning; Artificial Vector Fields Control.

Palavras-chaves: Robótica Móvel; Exploração por Fronteiras; Localização e Mapeamento; Planejamento de Caminhos; Controle por Campos Vetoriais.

ISSN: 2175-8905 DOI: 10.20906/sbai.v1i1.2702

# 1. INTRODUÇÃO

A utilização de robôs móveis para substituir a presença humana em ambientes perigosos tem atraído atenção da indústria e centros de pesquisa, possibilitando maior segurança para os profissionais envolvidos e diminuindo o risco de acidentes. Nesse sentido, o Instituto Tecnológico Vale (ITV) em parceria com a Universidade Federal de Minas Gerais (UFMG) vem desenvolvendo a plataforma robótica EspeleoRobô, com o objetivo de realizar missões de exploração e inspeção de ambientes confinados, como cavidades naturais, tubulações, e galerias de barragens (Azpurua et al., 2019).

Um dos principais objetivos durante missões de inspeção consiste em levantar um mapa representativo do ambiente percorrido. Esse mapa pode ser utilizado posteriormente para identificar modificações em plantas industriais ou problemas estruturais, além de servir de referência para a localização do próprio robô. A navegação de um robô em um ambiente a partir de um mapa consiste em uma tarefa relativamente mais simples, porém, isso requer a realização do mapeamento prévio, fornecendo informações exatas sobre obstáculos e espaços livres. Com isso, quando um robô é introduzido em um ambiente desconhecido, sua capacidade de navegação se torna limitada, necessitando de uma estratégia que possibilite a exploração desse ambiente de forma autônoma.

A exploração pode ser definida como a tarefa de navegar em um ambiente desconhecido, construindo um mapa de maneira iterativa, que é utilizado para a navegação subsequente. Mais recentemente, os avanços na exploração envolvem métodos melhorados de planejamento de caminhos (Reinhart et al., 2020) e cooperação entre robôs heterogêneos, como um helicóptero e um rover para exploração planetária (Sasaki et al., 2020) e subterrânea (Rouček et al., 2019; Ebadi et al., 2020). A exploração está diretamente ligada ao problema de localização e mapeamento simultâneos (em inglês Simultaenous Localization and Mapping, ou SLAM), que consiste em construir um mapa de um ambiente desconhecido e ao mesmo tempo estimar a localização do agente dentro desse mapa. Assim, soluções de Exploração normalmente envolvem o emprego de técnicas de SLAM. Além disso, para movimentar o robô dentro do mapa, são empregadas técnicas de planejamento de caminhos e controle de navegação.

Neste trabalho, foi desenvolvido um sistema para exploração autônoma de ambientes planares baseado em fronteiras, que são regiões localizadas no limite do ambiente conhecido pelo robô (Yamauchi, 1998). Ao se mover em direção às fronteiras, o robô gera um mapa do ambiente percorrido, até que todo o local de operação seja explorado. Para tal, é empregada uma estratégia de localização e mapeamento simultâneos por filtro de partículas (Thrun, 2002). Durante a exploração, as células de fronteira identificadas são agrupadas com base na distância entre elas. A exploração do ambiente pode ser realizada utilizando duas estratégias simples, priorizando a navegação até a fronteira

mais próxima ou até a fronteira com maior número de células. A movimentação do robô até as fronteiras de exploração é realizada utilizando um algoritmo de Dijkstra para planejamento de caminhos e um controle de navegação por campos vetoriais artificiais.

O sistema foi implementado como pacotes do *Robot Operating System* (ROS), um meta sistema operacional para desenvolvimento de *softwares* de robótica que reúne ferramentas, bibliotecas e convenções que simplificam a criação de programas para diferentes plataformas robóticas (Quigley et al., 2009). Os resultados obtidos por simulações e experimentos com um robô real demonstram a capacidade do sistema de criar mapas completos dos ambientes percorridos de maneira satisfatória.

Este artigo está organizado em 4 seções. A Seção 2 apresenta a metodologia desenvolvida para a exploração de ambientes, composta por estratégias de localização, mapeamento, detecção de fronteiras, planejamento de caminhos e controle de navegação. A Seção 3 apresenta resultados obtidos por simulação e experimentos realizados com um robô real de forma a validar a solução proposta. Por fim, a Seção 4 destaca as conclusões e propõe trabalhos futuros.

# 2. SISTEMA PROPOSTO PARA A EXPLORAÇÃO DE AMBIENTES PLANARES

O sistema proposto para a exploração de ambientes planares é representado na Figura 1. Este sistema possui como entradas a odometria das rodas do robô e a nuvem de pontos fornecida por um LiDAR (Light Detection And Ranging) planar instalado no robô. O módulo de SLAM utiliza essas informações para gerar um mapa do ambiente e uma estimativa da localização do robô em relação ao mapa à medida que o robô se movimenta no ambiente. A partir dessa mapa, as células de fronteira são detectadas e agrupadas. Em seguida, é feita a seleção da fronteira que será explorada, e as suas coordenadas são enviadas para o planejador de caminhos Dijkstra, que gera um caminho entre a posição atual do robô e o alvo. Esse caminho é utilizado como referência pelo controle por campos vetoriais responsável pela movimentação do EspeleoRobô. O processo de exploração acaba quando não há mais nenhuma fronteira a ser explorada, fornecendo como saída o mapa do ambiente explorado.

A implementação da metodologia consiste na integração das diferentes técnicas empregadas, utilizando algoritmos já disponibilizados como pacotes do ROS, juntamente com os nós de agrupamento de fronteiras, controle de navegação por campos vetoriais artificiais, e exploração implementados pelos próprios autores.

#### 2.1 Localização, mapeamento e detecção de fronteiras

A localização do robô e o mapeamento simultâneo do ambiente são realizados utilizando uma técnica de SLAM com Filtro de Partículas. Além disso, estratégias de detecção e agrupamento de fronteiras servem como base para a exploração do ambiente.

SLAM com Filtro de Partículas: A técnica de SLAM (Grisetti et al., 2007) utilizada neste trabalho é baseada em um Filtro de Partículas, que é um algoritmo bastante

ISSN: 2175-8905 DOI: 10.20906/sbai.v1i1.2702

<sup>\*</sup> Este trabalho foi parcialmente financiado pela Vale S.A. e Instituto Tecnológico Vale (ITV), Universidade Federal de Minas Gerais (UFMG), e o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

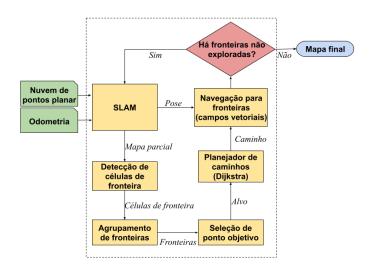


Figura 1. Representação gráfica do pacote de exploração de ambientes.

utilizado na Robótica Probabilística. O mapa m construído por esse algoritmo é representado no formato de uma grade de ocupação (Moravec and Elfes, 1985). Esta corresponde a uma representação matricial do ambiente no espaço 2D onde cada célula  $c_{ij}$  está associada a uma probabilidade  $p(c_{ij})$  de estar ocupada, possibilitando sua classificação como ocupada, livre ou desconhecida.

A ideia central dessa técnica é estimar a probabilidade a posteriori sobre os caminhos  $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$ , onde  $x_{1:t}$  é a trajetória do robô e m é o mapa. Essa estimativa é feita com base nos dados  $u_{1:t}$  de odometria e nas observações  $z_{1:t}$  provenientes do laser planar instalado no robô. Essa estimativa pode ser fatorizada como:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}|z_{1:t}, u_{1:t-1}).$$
(1)

O filtro de partículas é empregado para estimar os caminhos percorridos pelo robô (segunda parcela do lado direito da Equação 1). Cada partícula é uma candidata à solução do problema, possuindo um mapa associado a ela. O modelo de movimentação do robô permite calcular as próximas localizações de cada partícula. O mapa  $m_j$  associado a cada célula é atualizado pelas observações  $z_k$ . O filtro de partículas Rao-Blackwellized atualiza as partículas incrementalmente à medida que novas leituras da odometria das rodas ou do laser são disponibilizadas. Por meio do processo de amostragem e re-amostragem das partículas, após um certo número de iterações, o conjunto de partículas tende a convergir indicando a posição do robô e o mapa do ambiente.

Essa técnica de SLAM é disponibilizada em ROS no repositório: https://github.com/ros-perception/slam\_gmapping.

Detecção de fronteiras: As células de fronteira são identificadas no mapa fornecido pelo módulo de SLAM. Esse mapa tem o formato de grade de ocupação, onde cada célula representa uma posição x,y do mundo real. O método mais básico de detecção de fronteiras processa todas as células do mapa a cada iteração, classificando cada uma delas como uma célula de fronteira ou não. Em mapas que

possuem grandes dimensões e alta resolução, esse método pode ter um tempo de execução considerável, o que pode resultar em situações em que o robô fica parado após chegar em um objetivo, esperando o término da execução do algoritmo de detecção de fronteiras.

O método utilizado neste artigo é uma adaptação do Wavefront Frontier Detection (WFD) (Keidar and Kaminka, 2014), baseado em uma busca em grafo para a detecção de fronteiras. Esse método processa apenas as células conhecidas do mapa, não gastando tempo para verificar células desconhecidas que não possuem chance fazer parte de uma fronteira.

Inicialmente a célula do mapa que o robô está ocupando é inserida em uma estrutura de fila que determina a ordem de busca. A partir da célula de origem, uma busca é realizada nas células adjacentes. Caso alguma célula de fronteira seja encontrada, ela é adicionada ao conjunto de células de fronteira detectadas. Isso é feito para todas as células da região conhecida do mapa.

A Figura 2 mostra um exemplo do processo de detecção de fronteiras. Nessa imagem, as células de fronteira estão representadas em azul, os obstáculos em preto, as áreas livres em cinza claro, e as áreas desconhecidas em cinza escuro.

A implementação dessa técnica foi adaptada de: https://github.com/tpepels/turtlebot\_slam.

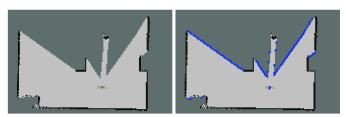


Figura 2. Detecção de pontos de fronteira em um mapa.

Agrupamento de fronteiras: Após a detecção das células de fronteira, é interessante agrupá-las em conjuntos que compõe uma fronteira isolada. Para isso, pode ser empregada uma metodologia para agrupamento das células de fronteira baseado na distância entre cada célula.

O Algoritmo 1 mostra o passo a passo da técnica adotada. Um vetor com as células de fronteira é recebido pelo algoritmo, e a primeira célula desse vetor é escolhida como referência (linha 5 do Algoritmo 1). É feita uma iteração no vetor de células, e a distância cartesiana de cada célula em relação à referência é calculada (linhas 9 e 10 do Algoritmo 1). Caso a distância seja menor que um valor d, a célula é adicionada à fronteira, o ponto de referência é atualizado com as coordenadas da nova célula, e a célula é removida do vetor (linhas 11 a 14 do Algoritmo 1). Esse processo é executado uma segunda vez para garantir o agrupamento de células de ambos os lados da célula inicial (linhas 17 a 28 do Algoritmo 1). Por fim, a fronteira é adicionada ao grupo de fronteiras frontiers que corresponde à saída do algoritmo (linhas 29 a 34 do Algoritmo 1). A Figura 3 mostra o processo completo de detecção e agrupamento de fronteiras. Os círculos vermelhos representam os pontos centrais de cada fronteira agrupada a partir das células de fronteira, em azul.

Outras técnicas mais comuns, como o próprio WFD, avaliam a conectividade entre células de fronteira a partir da adjacência entre elas. Porém, a ocorrência de informação ruidosa nas células pode gerar descontinuidades nas fronteiras agrupadas. Assim, a maior vantagem do Algoritmo 1 é permitir a avaliação da conectividade entre células a partir de um valor de distância parametrizado.

Esse algoritmo foi implementado em ROS e disponibilizado pelos autores em: https://github.com/ITVRoC/espeleo\_2d\_exploration.



Figura 3. Detecção e agrupamento de fronteiras em um mapa.

#### Algoritmo 1: Agrupamento de células de fronteira

```
Entradas: frontier_cells, //Vetor com células de fronteira.
          d, //Distância máxima para agrupamento de células
adjacentes
           min\_frontier\_cells //Número mínimo de células
agrupadas para criar uma nova fronteira.
        : frontiers //Vetor contendo os clusters de fronteiras.
2 Function ClusterFrontiers(frontier_points):
        create\_new\_frontier \longleftarrow TRUE
        while frontier\_cells.size() > 0 do
4
             if create_new_frontier then
5
                  ref \leftarrow (frontier\_cells[0]_x, frontier\_cells[0]_y)
6
                    //Seleciona a primeira célula como referência.
                  ref\_ini \longleftarrow ref
                  frontier\_cells.erase(0)
                  foreach cell \in frontier\_cells do
10
11
                       distance =
                         \sqrt{(ref_x - cell_x)^2 + (ref_y - cell_y)^2}
12
                       \mathbf{if}\ distance < d\ \mathbf{then}
13
                            current Frontier. append (cell) \\
14
                            15
                             frontier\_cells.erase(i)
                            break
16
17
                       end
18
                       if i == frontier\_cells.size() then
19
                            ref \longleftarrow ref\_ini
20
                            foreach cell_n \in frontier\_cells do
21
                                distance \gets
22
                                   \sqrt{(ref_x - cell_n.x)^2 + (ref_y - cell_n.y)^2}
                                 if distance < d then
23
                                      current Frontier. append (cell_n) \\
24
25
                                      ref \longleftarrow cell_n
                                      frontier\_cells.erase(j)
26
27
                                 end
                                      -j+1
28
                                j \leftarrow
29
                            end
                            if currentFrontier.size() >
30
                              min\_frontier\_cells then
                                frontiers.append(currentFrontier) \\
31
32
                            end
                            create\_new\_frontier \longleftarrow True
33
34
                            currentFrontier.clear()
35
                       \mathbf{end}
                             i + 1
36
37
                  end
38
             end
39
        end
        return frontiers
```

#### 2.2 Exploração, planejamento e navegação

Após identificadas as fronteiras, é preciso definir os pontos alvo a serem visitados pelo robô de forma a explorar todo o ambiente. Para tal, são empregadas estratégias de planejamento de caminho e controle de navegação.

Estratégia de exploração: O objetivo da estratégia de exploração é selecionar um ponto alvo dentre as fronteiras agrupadas e enviar esse ponto para o planejador de caminhos. As fronteiras representam regiões do mapa que possuem um alto potencial de fornecer novas informações sobre o ambiente à medida que o robô se aproxima delas, e portanto, elas se tornam pontos de interesse para a exploração.

O Algoritmo 2 mostra como essa estratégia funciona. As entradas são um vetor com as fronteiras agrupadas, e o algoritmo itera sobre esse vetor para encontrar a fronteira mais próxima à pose atual do robô, assim como a fronteira com maior número de pontos (linhas 1 a 13 do Algoritmo 2). O algoritmo seleciona o ponto alvo para exploração com base no modo de exploração escolhido: exploração pela fronteira mais próxima ou pela maior fronteira (linhas 14 a 19 do Algoritmo 2). Após a seleção da fronteira que será explorada, a célula da fronteira escolhida que está mais próxima do robô é escolhida como ponto alvo, e é enviada para o planejador de caminhos (linha 20 do Algoritmo 2). Por fim, o algoritmo espera o resultado do sistema de controle, com um timeout de 10 s (linha 21 do Algoritmo 2). Esse tempo é importante para evitar situações que bloqueiam a execução do algoritmo caso o ponto alvo selecionado seja de difícil acesso ou inalcançável pelo robô. Caso o timeout expire, uma nova fronteira é escolhida para exploração. Isso é feito até que não exista mais nenhuma fronteira no ambiente explorado.

Além disso, no início da exploração, o algoritmo permite que o robô gire em torno do seu próprio eixo, para maximizar o ganho de informação inicial sobre o ambiente em questão.

A estratégia de exploração também foi disponibilizada pelos autores em: https://github.com/ITVRoC/espeleo\_2d\_exploration.

Algoritmo de Dijkstra para planejamento de caminhos: Para realizar o planejamento de caminho entre a posição atual do robô e a posição alvo selecionada, é utilizado o algoritmo de Dijkstra. Este algoritmo utiliza uma função de custo dada por f(n) = g(n), onde g(n) avalia o custo do nó inicial até o atual (Choset et al., 2005), onde o custo pode ser dado por diferentes métricas, por exemplo a menor distância euclidiana. Assim, o método realiza a expansão do caminho de menor custo até que a posição final seja encontrada.

O algoritmo de Dijkstra realiza a classificação dos nós de um grafo durante a exploração do mesmo, possuindo como critério uma função que analisa o custo do nó de origem até o nó explorado. Cada nó do grafo corresponde à uma célula da grade de ocupação, já as arestas estão associadas à distância euclidiana entre duas células adjacentes. O algoritmo utiliza uma fila de prioridade para armazenar os nós candidatos a expansão. O algoritmo mantém um

# Algoritmo 2: Exploração por fronteiras.

```
Entradas: frontiers, //Vetor de fronteiras.
             p, //Pose atual do robô em relação ao mapa.
             mode //Modo de exploração.
1 if frontiers.size() \neq 0 then
         closestFrontier \longleftarrow frontiers[0]
         largestFrontier \longleftarrow frontiers[0]
3
 4
            \sqrt{(p_x - closestFrontier_x)^2 + (p_y - closestFrontier_y)^2}
         \textbf{for each}\ frontier \in frontiers\ \textbf{do}
               \begin{aligned} distance &\longleftarrow \sqrt{(p_x - frontier_x)^2 + (p_y - frontier_y)^2} \\ \text{if } distance &\leq d_c \text{ then} \end{aligned}
 6
 7
                    closestFrontier \longleftarrow frontier
 8
 9
                     end
10
               \label{eq:frontier} \textbf{if} \ [frontier.size() > largestFrontier.size() \ \textbf{then} \\
11
                    largestFrontier \longleftarrow frontier
12
13
         end
14
         if \ mode == "largest" \ then
15
               goal \longleftarrow getClosestCell(largestFrontier)
16
17
         end
         if mode == "closest" then
18
19
              goal \leftarrow getClosestCell(closestFrontier)
20
21
          sendGoal(goal)
          waitForResult(10)
22
23 end
```

conjunto C de nós os quais já tiveram o custo do menor caminho já determinado, mais um conjunto O de nós ainda não visitados. O restante dos nós são visitados utilizando a estimativa de menor caminho e então adicionados à C (Cormen et al., 2001).

#### Algoritmo 3: Algoritmo de Dijkstra

```
Entradas: x_{init} // posição inicial do robô
             x_{final} // posição alvo
Saídas : path // caminho entre os pontos inicial e final
1 \ O \leftarrow O \cup \{x_{init}\}
   n_{melhor} \leftarrow x_{init}
3 repeat
          if n_{melhor} \notin C then
                C \leftarrow C \cup \{n_{melhor}\}
                if x \notin C \ \forall \ x \ vizinhos \ de \ n_{melhor} then
                     O \leftarrow O \cup \{x\}
 7
                end
 8
10
          n_{melhor} \leftarrow getBest(O) \ \forall \ f(n_{melhor}) \leq f(n), \forall \ n \in O
11 until O estar vazio ou x<sub>final</sub> ∈ C;
12 return path
```

O pseudo-código do Dijkstra é apresentado no Algoritmo 3. São passadas como entrada para o algoritmo a posição inicial e a posição alvo. Primeiro, a posição inicial do robô  $x_{init}$  é adicionada ao grupo dos nós não visitados O, e a variável  $n_{melhor}$  recebe o valor de  $x_{init}$  (linhas 1 e 2 do Algoritmo 3). Logo em seguida, o algoritmo entra em um loop onde inicialmente é verificado se o nó atual está no conjunto C; caso não esteja, ele é imediatamente adicionado à C e então todos os seus vizinhos são adicionados ao conjunto O de nós a serem visitados (linhas 5 a 9). Em seguida, é escolhido o nó de menor custo que esteja em O para ser visitado de acordo com a função de custo f(n) (linha 10). Esse procedimento é repetido até que o ponto de objetivo  $x_{final}$  esteja no conjunto C ou então até O ficar vazio.

O planejador com algoritmo de Dijkstra faz parte do ROS Navigation Stack, disponibilizado em: https://github.com/ros-planning/navigation.

Controle de navegação por campos vetoriais: Para a navegação do robô móvel, é possível utilizar uma técnica de controle por campos vetoriais. Tal técnica define uma velocidade de referência composta por uma componente convergente e uma tangencial. A componente convergente comanda o robô até um caminho de referência, e a componente tangencial guia o robô ao longo do caminho. A estrutura de campo vetorial é similar ao que é apresentado em (Gonçalves et al., 2010). Tomando como referência uma curva de nível  $\alpha(\mathbf{p}_R) = 0$ , o campo vetorial é definido por:

$$\mathbf{F}(\mathbf{p}) = v_r \left( G(\alpha) \frac{\nabla \alpha}{\|\nabla \alpha\|} + H(\alpha) \frac{\nabla_H \alpha}{\|\nabla \alpha\|} \right), \tag{2}$$

onde  $v_r$  é uma velocidade de referência para o robô,  $G(\alpha) = -(2/\pi) \operatorname{atan}(k_f \alpha)$  com ganho  $k_f$  positivo, e  $H = \sqrt{1 - G^2}$ . Além disso,  $\nabla \alpha$  é o gradiente de  $\alpha$  e  $\nabla_H \alpha$  é o campo Hamiltoniano de  $\alpha$ , ou seja, o vetor  $\nabla \alpha$  rotacionado em  $90^\circ$ .

Para a exploração, a curva é aberta e definida por uma sequência de pontos, gerados pelo planejador Dijkstra. A partir desses pontos, uma função  $\alpha$  é aproximada usando o método numérico descrito em (Gonçalves et al., 2010). O algoritmo de navegação por campos vetoriais recebe a sequência de pontos e a pose atual do robô.

Essa técnica de controle por campos vetoriais gera uma velocidade de referência para o robô. Entretanto, considerando robôs móveis com restrições não holonômicas, por exemplo o EspeleoRobô, que recebem comandos de velocidade linear v e angular  $\omega$ , é possível empregar uma estratégia de  $Feedback\ Linearization$  proposta em (Siciliano et al., 2010), de forma a calcular as velocidades de comando tal que um ponto B localizado a uma certa distância  $d_{feedback}$  à frente do robô alcance a velocidade desejada F. Sendo  $F_x$  e  $F_y$  as velocidades lineares de referência do ponto B nos eixos x e y, é possível definir as seguintes velocidades de comando para o robô:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & -\cos(\psi) \\ \overline{d_{feedback}} & \overline{d_{feedback}} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}.$$
 (3)

Essa técnica de controle foi implementada em ROS pelos autores e está disponível em: https://github.com/ITVRoC/espeleo\_control.

# 3. SIMULAÇÕES E EXPERIMENTOS

Para validar o sistema de exploração, foram realizadas simulações e experimentos com um robô móvel. No primeiro caso, foi utilizado um simulador do EspeleoRobô (Figura 4), desenvolvido utilizando o *CoppeliaSim* junto com o ROS (Cid et al., 2020). Já os experimentos foram realizados utilizando um robô iRobot Create (Figura 5) equipado com um *laser* Hokuyo URG-O4LX-UG01 e um netbook com processador Intel Atom e sistema operacional Lubuntu 16.04 executando o ROS. Além disso, foi utilizada uma máquina em rede dedicada, com processador Intel 4ª geração e 8GB de memória RAM. Essa máquina também

roda Ubuntu 16.04 e ROS para processamento *online* dos dados provenientes do robô e do *laser* e envio de comandos de velocidade para o robô, sendo o netbook utilizado apenas para a coleta e transmissão desses dados.

Os vídeos das simulações e experimentos estão disponíveis online.  $^{1}\,$ 



Figura 4. Modelo virtual do EspeleoRobô no CoppeliaSim utilizado para validação do sistema de exploração.



Figura 5. iRobot Create utilizado para validação do sistema de exploração.

Para avaliar o desempenho do sistema, algumas métricas foram definidas para comparar as duas estratégias de exploração propostas. A primeira delas é a distância total percorrida pelo robô durante a exploração, calculada como:

$$d_{total} = \sum_{i=0}^{N} \sqrt{(p_x^i - p_x^{i-1})^2 + (p_y^i - p_y^{i-1})^2},$$
 (4)

onde  $p_x^1,...,p_x^N$  e  $p_y^1,...,p_y^N$  são as coordenadas x e y do robô durante o experimento. Essas posições são amostradas da estimativa da posição do robô fornecidas pelo módulo de SLAM.

Já a segunda métrica consiste no número de ações do robô, que é incrementado sempre que uma nova posição alvo é enviada para o sistema de navegação.

Os testes foram realizados em dois cenários distintos, sendo o primeiro um apartamento com quatro cômodos, e o segundo um ambiente urbano utilizado na DARPA Subterranean Challenge,  $^2$  uma competição proposta pela agência de defesa americana para promover a utilização de robôs móveis na exploração de ambientes subterrâneos.

Além disso, os mapas criados possuem resolução de  $0.05 \,\mathrm{m/c\acute{e}lula}$  e tamanho fixo de  $40 \,\mathrm{x} 40 \,\mathrm{m}$ .

	Fronteira mais próxima	
Cenário	Núm. Comandos	Distância [m]
Apartamento simulado	$\textbf{8,33}\pm\textbf{1,24}$	$22,\!81\pm1,\!92$
Apartamento real	8	21,57
	Maior fronteira	
1		1100110
Cenário	Núm. Comandos	Distância [m]
Cenário Apartamento simulado		

Tabela 1. Métricas calculadas durante a exploração do apartamento.

#### 3.1 Exploração do apartamento

Inicialmente o sistema de exploração foi verificado num ambiente simples, que corresponde a um apartamento de quatro cômodos. Para validar os resultados obtidos no ambiente real e comparar o comportamento do sistema, foi criado um modelo virtual do apartamento explorado, conforme ilustrado na Figura 6. Cada uma das estratégias foi verificada realizando três simulações e um experimento de exploração do apartamento com um robô real.

A Tabela 1 apresenta as métricas de desempenho obtidas pelas diferentes estratégias de exploração. É possível notar que a exploração pela fronteira mais próxima teve um desempenho melhor tanto em simulação quanto nos experimentos reais, acarretando em menores distâncias percorridas e um menor número de ações realizadas pelo robô.



Figura 6. Modelo virtual do apartamento.

A Figura 7 mostra os mapas construídos durante as simulações (mapas de cima) e experimentos (mapas de baixo), além da sequencia de pontos alvo e dos caminhos realizados pelos robôs utilizando as diferentes estratégias de exploração. A Figura 8 mostra a sequência de posições que o robô percorreu durante o experimento de exploração pela fronteira mais próxima. É possível notar que o sistema obteve sucesso ao construir mapas completos tanto no ambiente real quanto cenário simulado. Nesse cenário, a diferença entre os modos de operação fica clara. Ao explorar o ambiente com base no número de células de cada fronteira, o robô visitou primeiro o cômodo no final do corredor, tendo que voltar pelo corredor para visitar o cômodo anterior. Por outro lado, a exploração pela fronteira mais próxima resultou em um caminho mais

 $<sup>^{1}\ \</sup>mathtt{https://github.com/lucca-leao/espeleo\_2d\_exploration}$ 

<sup>&</sup>lt;sup>2</sup> https://www.subtchallenge.com/

lógico, no qual o robô visitou todos os cômodos em sequência, sem precisar voltar atrás.

É interessante notar que o robô nem sempre alcança os pontos alvos. Isso acontece pois o mapa é atualizado continuamente à medida que o robô se aproxima da fronteira escolhida, sendo possível, em muitos casos, eliminar uma fronteira antes mesmo que o robô chegue de fato à posição alvo. Tanto nas simulação quanto nos experimentos reais, a exploração pela fronteira mais próxima gerou uma sequência de posições alvos mais eficiente, fazendo o robô percorrer uma distância menor.



Figura 7. Mapas do apartamento gerados em simulação e experimentos reais, junto com os percursos realizados pelo robô.

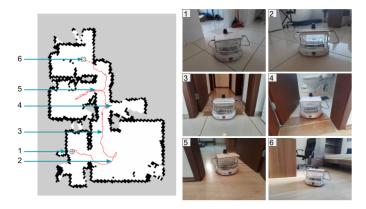


Figura 8. Imagens do experimento realizado com a exploração pela fronteira mais próxima.

Apesar de alguns ruídos apresentados no mapa gerado durante o experimento real, é possível observar na Tabela 1 e Figura 7 a grande semelhança entre os resultados obtidos por simulação e experimentos, demonstrando a execução adequada do sistema de exploração proposto tanto em sistemas virtuais quanto embarcado em robôs móveis reais.

Fronteira mais próxima		
Núm. Comandos	Distância [m]	
$10{,}33\pm0{,}94$	$60,\!14\pm2,\!31$	
Maior fronteira		
Núm. Comandos	Distância [m]	

Tabela 2. Métricas calculadas durante a exploração do ambiente urbano da competição da DARPA.

3.2~Exploração~do~ambiente~urbano~da~competição~da~DARPA

O segundo cenário de testes é um galpão industrial utilizado na competição DARPA Subterranean Challenge, ilustrado na Figura 9. Este corresponde a um ambiente mais complexo que o apartamento explorado anteriormente, com área  $3 \times$  maior, e com uma região inicial com diversos obstáculos conectada por um corredor a uma área mais ampla e aberta. Cada uma das estratégias foi verificada realizando três simulações de exploração do ambiente urbano.

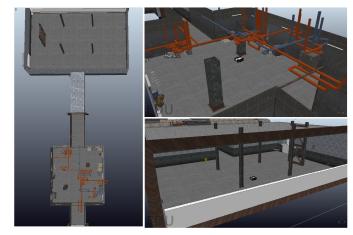


Figura 9. Modelo virtual do ambiente urbano da competição DARPA SubT.

A Tabela 2 apresenta as métricas de desempenho obtidas pelas diferentes estratégias durante a exploração do cenário virtual. Neste caso, a distância percorrida pelo robô explorando o ambiente com base nas fronteiras mais próximas foi 13% menor que a distância percorrida utilizando a estratégia de exploração de fronteiras com maior número de pontos.

A Figura 10 apresenta os mapas construídos durante as simulações, além da sequencia de pontos alvo e dos caminhos realizados pelos robôs utilizando as diferentes estratégias de exploração. Nesse cenário, a exploração pela maior fronteira também teve um comportamento pior. Ao alcançar as fronteiras na área mais ampla, o robô voltou para explorar uma pequena fronteira na área inicial do ambiente. Assim como no apartamento, a exploração do ambiente urbano pela fronteira mais próxima gerou uma sequência de posições alvos mais eficiente, fazendo o robô percorrer uma distância menor.

## 4. CONCLUSÕES E TRABALHOS FUTUROS

Este artigo apresentou um sistema de exploração autônoma de ambientes planares baseado em fronteiras. A



Figura 10. Mapas do ambiente urbano da competição DARPA SubT gerados em simulação, junto com os percursos realizados pelo robô.

solução proposta emprega diferentes técnicas comumente utilizadas na robótica, incluindo o SLAM com Filtro de Partículas, a detecção de fronteiras, o planejamento de caminhos com o algoritmo de Dijkstra, e controle de navegação por campos vetoriais artificiais.

Duas estratégias simples de exploração são propostas, priorizando a navegação até a fronteira mais próxima ou até a fronteira com maior número de células. Em todos as as simulações e experimentos realizados, a exploração pela fronteira mais próxima obteve um melhor desempenho, tanto em distância percorrida quanto em número de ações realizadas pelo robô.

Apesar dos desenvolvimentos serem motivados pela aplicação no EspeleoRobô, o sistema de exploração proposto é generalista o suficiente para ser utilizado por qualquer robô móvel com sistema de odometria das rodas e sensor LiDAR 2D. Isso fica ilustrado pelos resultados experimentais obtidos com o robô iRobot Create.

Como trabalhos futuros, o sistema de exploração será avaliado em cenários mais complexos incluindo labirintos, ajustando os parâmetros de min\_frontier\_cells e d para aumentar a robustez durante a identificação de fronteiras. Estratégias de exploração mais sofisticadas serão avaliadas, selecionando a próxima fronteira a ser visitada considerando a relação entre o ganho de informação e o custo de deslocamento do robô. Além disso, estão previstas também a utilização de um LiDAR 3D para realizar o mapeamento volumétrico do ambiente, e a investigação de técnicas de detecção de fronteiras e mapeamento tridimensional de ambientes irregulares.

### REFERÊNCIAS

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 International Conference on Advanced Robotics (ICAR). IEEE. doi: 10.1109/icar46387.2019.8981627. URL https://doi.org/10.1109/icar46387.2019.8981627.

Choset, H.M., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S., and Arkin, R.C. (2005). Principles of robot motion: theory, algorithms, and implementation. MIT press.

Cid, A., Nazário, M., Sathler, M., Martins, F., Domingues, J., Delunardo, M., Alves, P., Teotônio, R., Barros, L.G., Rezende, A., et al. (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. IEEE.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2001). Introduction to algorithms second edition. *The Knuth-Morris-Pratt Algorithm*.

Ebadi, K., Chang, Y., Palieri, M., Stephens, A., Hatteland, A., Heiden, E., Thakur, A., Funabiki, N., Morrell, B., Wood, S., et al. (2020). Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. In 2020 IEEE International Conference on Robotics and Automation (ICRA), 80–86. IEEE.

Gonçalves, V.M., Pimenta, L.C., Maia, C.A., Dutra, B.C., and Pereira, G.A. (2010). Vector fields for robot navigation along time-varying curves in *n*-dimensions. *IEEE Transactions on Robotics*, 26(4), 647–659.

Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE transactions on Robotics*, 23(1), 34–46.

Keidar, M. and Kaminka, G.A. (2014). Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2), 215–236.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, 116–121. IEEE.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 5. Kobe, Japan.

Reinhart, R., Dang, T., Hand, E., Papachristos, C., and Alexis, K. (2020). Learning-based path planning for autonomous exploration of subterranean environments. In 2020 IEEE International Conference on Robotics and Automation (ICRA), 1215–1221. IEEE.

Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanskỳ, V., Heřt, D., Petrlík, M., Báča, T., Spurnỳ, V., et al. (2019). DARPA Subterranean Challenge: Multirobotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systems*, 274–290. Springer.

Sasaki, T., Otsu, K., Thakker, R., Haesaert, S., and Aghamohammadi, A.a. (2020). Where to map? iterative mars helicopter-rover path planning for long-range autonomous exploration.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). Robotics: modelling, planning and control. Springer Science & Business Media.

Thrun, S. (2002). Probabilistic robotics. Communications of the ACM, 45(3), 52–57.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, 47–53.