Exploração Autônoma com Múltiplos Robôs Baseada em Fronteiras e Planejamento RRT*

Victor R. F. Miranda [*] **Armando A. Neto** [**] **Gustavo M. Freitas** [***]
Israel Amaral [*] **Luciano C. A. Pimenta** [**] **Leonardo A. Mozelli** [**]

[*] *Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil, victormrfm@ufmg.br*
[**] *Departamento de Engenharia Eletrônica, UFMG, Belo Horizonte, Brasil, (aaneto,lucpim,mozelli)@cpdee.ufmg.br*
[***] *Departamento de Engenharia Elétrica, UFMG, Belo Horizonte, Brasil, gustavomfreitas@ufmg.br*

Abstract: The risks in exploring unknown environments have attracted interest in using mobile robots. In addition, the coordinated and cooperative use of multiple robots to perform these tasks can provide better efficiency in exploration. In this context, this work presents the development of an exploration system with multiple robots for mapping, using a boundary selection method based on the exchange of information among robots. In addition, a planner based on the rapidly-exploring random tree (RRT*) determines the path to the chosen frontier point, carrying out replanning if necessary during the route, always aiming to go to strategic frontier points and avoiding obstacles that may appear along the way. Simulated experiments are presented comparing the use of only one robot for exploration to the use of more than one.

Resumo: Os riscos que envolvem as operações de exploração de ambientes desconhecidos têm atraído interesse para o uso de robôs móveis. Além disso, o uso coordenado e cooperativo de múltiplos robôs para execução dessas tarefas pode proporcionar uma maior eficiência na exploração. Nesse contexto, o presente trabalho apresenta o desenvolvimento de um sistema de exploração com múltiplos robôs para mapeamento, utilizando um método de seleção de fronteiras baseado na troca de informação entre os robôs. Além disso, um planejador baseado no *rapidly-exploring random tree* (RRT*) determina o caminho até o ponto de fronteira escolhido, realizando replanejamentos sempre que necessário durante o percurso, visando seguir sempre para pontos de fronteira estratégicos e evitando obstáculos que venham a surgir durante o percurso. Experimentos em simulador são apresentados comparando o uso de apenas um robô para exploração ao uso de mais de um.

Keywords: Autonomous exploration; mobile robotics; multiagent systems; multiple robots; RRT

Palavras-chaves: Exploração autônoma; robótica móvel; sistemas multiagentes; múltiplos robôs; RRT

1. INTRODUÇÃO

Os avanços tecnológicos nos últimos anos influenciam de forma crescente o uso da Robótica Móvel no dia a dia da indústria e da sociedade como um todo. Robôs móveis terrestres têm sido utilizados em aplicações como transporte (Miranda et al., 2021), exploração (Azpúrua et al., 2021), monitoramento (Vaidis and Otis, 2020) e até no auxílio de tarefas domésticas(Pathmakumar et al., 2021; Mir-Nasiri et al., 2018). O uso dessas tecnologias pode proporcionar diversos benefícios com respeito à segurança, desempenho e eficiência em geral.

No contexto de exploração e mapeamento de ambientes desconhecidos, as operações podem ocorrer em locais confinados, com risco de desabamento, na presença de animais peçonhentos, entre outros fatores que podem oferecer risco à vida humana. Neste caso, o uso da robótica móvel autônoma auxilia na segurança e agilidade dessas tarefas. Além disso, visando maximizar os resultados das operações, diversas pesquisas têm abordado a cooperação entre múltiplos robôs autônomos para a realização de tarefas de exploração (de Hoog et al., 2009; Qin et al., 2019; Hu et al., 2020; Corah et al., 2019; Kulkarni et al., 2022).

O desenvolvimento de sistemas de navegação autônoma para robôs móveis envolve uma integração de técnicas de localização, planejamento de movimento e controle. Considerando um caso de operação multi-agente, cada um deve ter o seu sistema de navegação próprio, além de estabelecer comunicação com os demais, por meio de uma topologia determinada para tomadas de decisão durante a execução da tarefa.

O presente trabalho compreende o desenvolvimento de um sistema de exploração autônomo baseado em fronteiras

para ambientes desconhecidos, utilizando múltiplos robôs móveis para construção de um mapa 3D. O método consiste em determinar o destino a ser seguido por cada um dos robôs a partir de uma análise das fronteiras do mapa de ocupação local. Uma heurística é utilizada por cada robô para determinar o melhor ponto a ser alcançado, de forma online, a medida que o mapa é explorado, baseando-se nas posições de cada agente no ambiente e nas fronteiras encontradas. Dessa forma, é possível maximizar a exploração do ambiente desconhecido e evitar ambiguidade nas rotas.

O método de planejamento adotado utiliza um algoritmo por amostragem *rapidly-exploring random tree* (RRT*) (Karaman and Frazzoli, 2011) para definir caminhos otimizados minimizando o custo de distância entre a posição atual do robô e o ponto de destino, evitando colisões com obstáculos. Além disso, é proposta uma forma de replanejamento ao longo do caminho e construção do mapa, para que o robô sempre siga para o melhor ponto de fronteira e também evite colisões com novos obstáculos que surgirem.

A localização do robô é dada pela odometria das rodas e o mapa do ambiente é construído gradativamente a partir da nuvem de pontos de um sensor *Light Detection And Ranging* (LiDAR) utilizando o software OctoMap (Hornung et al., 2013). O algoritmo salva a detecção 3D do LiDAR e combina com os dados da odometria para gerar um mapa de ocupação 2D do ambiente e dos obstáculos detectados em 3D. Esse mapa de ocupação 2D é utilizado para determinar o espaço livre para movimento durante o planejamento de caminhos.

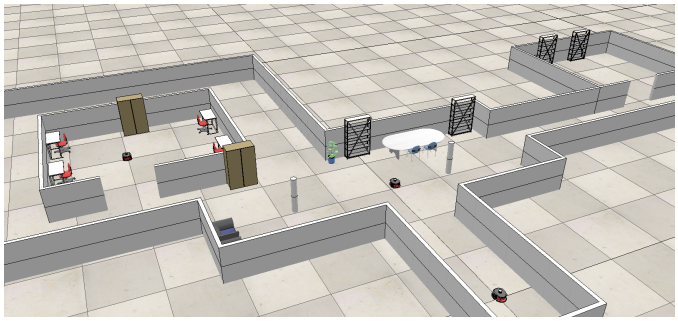

Figura 1. Ambiente de simulação CoppeliaSim.

A estratégia de controle compreende seguir o caminho gerado pelo planejador, computando numericamente o gradiente do erro entre a posição do robô e o caminho discretizado. Resultados de simulação obtidos em um ambiente CoppeliaSim [1] (Figura 1) com integração do ROS permitem avaliar o desempenho do método desenvolvido e comparar o ganho com o uso de múltiplos robôs para a operação de exploração.

O restante do artigo está estruturado da seguinte forma: na Seção 2 são apresentadas as técnicas e algoritmos utilizados para determinação dos pontos de fronteira, planejamento de caminho e controle na operação de exploração. Na Seção 3, são apresentados os resultados dos experimentos realizados no CoppeliaSim e uma análise comparativa do uso de múltiplos robôs. Por fim, na Seção 4 são discutidas as conclusões e perspectivas futuras.

[1] https://www.coppeliarobotics.com/

2. METODOLOGIA

O método desenvolvido apresenta 4 módulos principais, descritos nessa seção, e o algoritmo de mapeamento, conforme ilustra a Figura 2. O sistema opera de forma separada em cada robô atuando no ambiente.

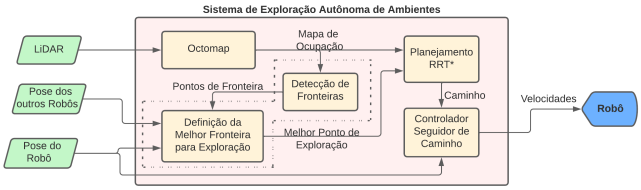

Figura 2. Diagrama de blocos ilustrativo do funcionamento do sistema de exploração.

2.1 Localização e Mapeamento

O método de localização e mapeamento utilizado é baseado na odometria das rodas por meio de *encoders* e um sensor do tipo LiDAR. A odometria fornece dados de posição do robô no mundo e informações de velocidade linear (v) e angular (ω), enquanto o LiDAR fornece uma nuvem de pontos 3D dos objetos detectados no ambiente.

O mapeamento é feito por meio da nuvem de pontos 3D, utilizando o algoritmo Octomap (Hornung et al., 2013). Este é um algoritmo de código livre que utiliza *octrees* e uma abordagem de fusão probabilística para determinar o mapa do ambiente discriminando espaços ocupados, livres e desconhecidos. Os dados são gerados a partir de uma informação de posição do sensor LiDAR e da sua nuvem de pontos do ambiente. O mapa é construído, gradualmente, de acordo com o deslocamento do sensor no ambiente e o aumento da nuvem de pontos detectados.

A Figura 3 apresenta um exemplo do mapa gerado pelo Octomap no ambiente simulado, contendo o mapa de ocupação 2D e os obstáculos em 3D. Esse mapa de ocupação 2D é discretizado a partir de uma determinada resolução em uma matriz, onde cada elemento representa um espaço livre, ocupado ou desconhecido do mapa. Dessa forma é possível saber o estado de um determinado ponto do mapa e ainda obter a sua localização em 2D (x,y). Assim, esse mapa de ocupação serve de entrada do algoritmo de planejamento de caminhos, que determina a sequência de pontos passando por espaços livres até o destino.

O mapa é gerado localmente por cada um dos robôs. A união dos mapas locais em um único mapa global é feita utilizando transformação dos dados locais para um referencial global, considerando as matrizes de transformação homogênea da posição do sensor no referencial do robô e também do robô no referencial do mundo.

2.2 Determinação de Fronteiras para Exploração

Uma parte importante em um sistema de exploração para ambientes desconhecidos é determinar pontos estratégicos a serem alcançados, com o objetivo de maximizar a exploração e reduzir o tempo gasto. Diversas estratégias são apresentadas na literatura (Schmidt et al., 2006; Qin et al.,

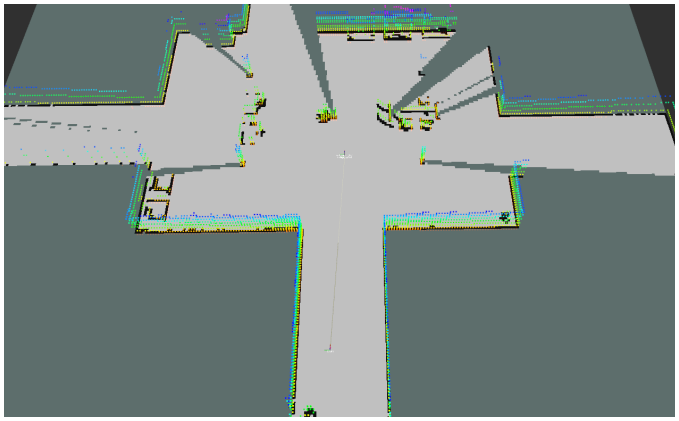

Figura 3. Exemplo de construção de um mapa a partir do Octomap.

2019; Niroui et al., 2019; Cieslewski et al., 2017), cada uma com sua particularidade e vantagem em relação a outra. Uma das formas mais simples é a determinação de um ponto aleatório no mapa conhecido. Outra forma muito utilizada é a navegação por rotas fixas (Sim and Dudek, 2003), mas é uma estratégia que apresenta problemas de revisitação a locais já explorados.

Um método muito abordado consiste em determinar os destinos de interesse a partir da identificação dos pontos de fronteira do mapa explorado (Yamauchi, 1997). Essa estratégia seleciona os pontos do mapa que fazem fronteira entre o setor de espaço livre e o setor desconhecido no mapa global de ocupação. A determinação do ponto a ser seguido, parte de problemas de otimização, utilizando um determinado custo.

Baseado em métodos de determinação por fronteiras (Yamauchi, 1997; Colares and Chaimowicz, 2016; Lu et al., 2020), criamos um algoritmo para determinar um ponto de exploração especifico para cada robô no ambiente. Esse algoritmo determina as fronteiras como sendo os pontos no mapa que, além de estarem entre um ponto de espaço livre e outro de setor desconhecido, devem pertencer ao espaço livre e estar a uma determinada distância dos obstáculos conhecidos. Os pontos vermelhos na Figura 4 ilustram todos os pontos de fronteira determinados utilizando o método descrito.

A partir do vetor de pontos de fronteira candidatos, a determinação do destino para cada um dos agentes passa por encontrar o ponto que minimiza a distância para a posição do robô e que maximiza a distância para os outros robôs no ambiente. Isso evita planejamento de caminhos muito longos, o que reduz o custo computacional, e também evita visitações repetidas a locais no ambiente por mais de um robô. Dessa forma, propomos um método para cada robô encontrar o seu ponto de destino p_f de forma independente, considerando como base de calculo a informação de posição dos outros agentes envolvidos na tarefa, da seguinte forma:

$$p_f = \min\left(\frac{D_{r_i}}{\sum_{j=1}^{m} D_{a_{i,j}}}\right), \; i = 1, 2, ..., N \qquad (1)$$

onde N é o número de fronteiras candidatas, D_{r_i} é a distância Euclidiana da fronteira i para a posição do robô em análise e $D_{a_{i,j}}$ a distância Euclidiana da fronteira i para o outro robôs j dentre os m presentes no sistema, excluindo o atual em análise.

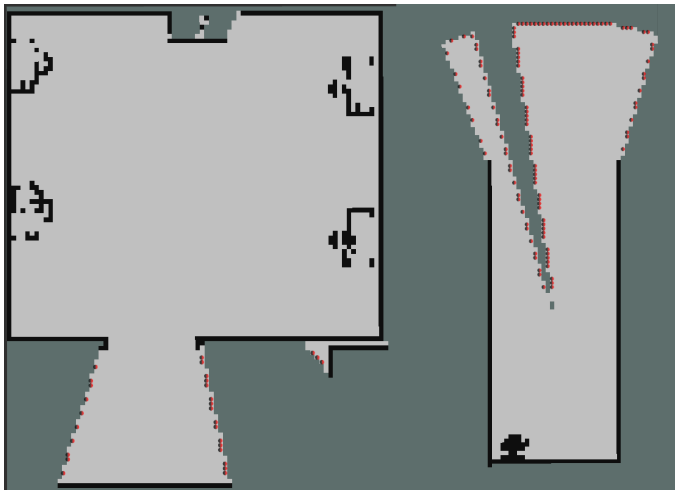

Figura 4. Seleção dos pontos de fronteira candidatos (em vermelho) no mapa de ocupação.

*2.3 Planejador RRT**

O planejamento do caminho entre a posição do robô até um ponto de destino p_f é feito de forma independente por cada robô, utilizando um novo algoritmo RRT* proposto, que realiza replanejamentos no caminho já planejado inicialmente, com o objetivo de minimizar esforço computacional.

A identificação dos obstáculos é feita a partir do mapa de ocupação gerado pelo Octomap, sendo possível definir posições x e y dos obstáculos como proibidas no planejamento. Dessa forma, um vetor contendo a posição de cada obstáculo e também a resolução da discretização do mapa servem de entrada para o algoritmo que é recalculado a medida que o mapa é explorado.

A configuração inicial do robô (posição x, y e orientação θ) é tomada como a raiz da árvore q_0, que irá se expandir de forma iterativa até encontrar a configuração final desejada. Portanto, o critério de parada assumido é uma distância mínima até o destino, o que resultará em um planejamento sub-ótimo.

A cada iteração, uma configuração inicial aleatória q_{rand} é definida dentro do espaço livre do mapa. Na sequência, uma busca é feita dentre os nós já pertencentes à árvore pela configuração q_{near} de menor distância Euclidiana para q_{rand}. Assim, um incremento dado por δ é feito na direção de q_{near} para q_{rand} e se o ponto encontrado q_{new} for livre de obstáculos, ele é adicionado à árvore. Devido a não holonomia do robô e para obtenção de caminhos mais suaves, o incremento δ não é feito em linha reta na direção de q_{rand}, como no algoritmo RRT original. É considerado o modelo cinemático discreto do robô para propagar os estados e determinar uma sequência de pontos a serem seguidos entre q_{near} e q_{new}. A equação dessa propagação é dada por:

$$\begin{aligned} x[k] &= x[k-1] + u_V \cos(\theta[k-1])\triangle t, \\ y[k] &= y[k-1] + u_V \sin(\theta[k-1])\triangle t, \\ \theta[k] &= \theta[k-1] + u_V \tan(u_\theta)\triangle t, \end{aligned} \qquad (2)$$

onde u_V é a velocidade desejada para o robô na formação da trajetória, u_θ o ângulo de curvatura, $\triangle t$ um passo de tempo, $x[k]$ e $y[k]$ posição em metros e $\theta[k]$ orientação no instante atual.

Cada nó acrescentado na árvore possui um custo associado, que é dado pelo soma entre o custo do nó anterior e à distância percorrida até o atual, da seguinte forma:

$$Cost(q_{new}) = Cost(q_{near}) + Dist(q_{new}, q_{near}), \quad (3)$$

onde $Cost(.)$ é o custo e $Dist(.)$ representa o cálculo da distância Euclidiana.

Uma análise é feita para encontrar os nós da árvore que estão a uma distância δ de q_{new} e buscar as arestas de menor custo da raiz até q_{new}. Se as arestas atuais forem as de menor custo, o algoritmo segue para a próxima iteração, senão a árvore é reconstruída para conectar q_{new} a um nó pai de menor custo. Todo esse processo é feito sempre realizando checagem de colisão com obstáculos, que pode eliminar um nó da árvore. Uma colisão é acusada quando o ponto está a uma distância dada por duas vezes a resolução do mapa de ocupação de qualquer obstáculo.

O algoritmo 1 apresenta um pseudo-código desse processo: a função $RAND()$ determina uma configuração aleatória no espaço livre C_{free}; a função $NEAR(T, q)$ encontra o nó pertencente a árvore mais próximo de q; $STEP(q_1, q_2)$ determina uma configuração a uma distância de δ de q_1 para q_2 e um caminho até essa configuração de acordo com a equação (2); $CollisionFree(q_1, q_2, \mathbf{O})$ testa se existe colisão no caminho entre q_1 e q_2, avaliando a distância para todos os obstáculos em $\mathbf{O}$; $FindNear(q)$ encontra os nós da árvore dentro de um raio δ com centro em q; a função $BestParent(q, \mathbf{Q})$ encontra o nó que gera menor custo até q dentre os listados em $\mathbf{Q}$ e o atualiza como nó pai de q, reconstruindo a árvore com a função $STEP(q_1, q_2)$, se necessário; a função $REWIRE(q, \mathbf{Q})$ avalia para cada nó em $\mathbf{Q}$ se existe um custo menor para alcançá-los partindo de q, em caso positivo conecta um vértice entre os nós, deixando o caminho mais suave; a função $DistToGoal(q, T)$ checa se existem nós pertencentes à árvore que estejam a uma distância mínima de q, avalia se existem obstáculos no caminho desses nós até q e checa qual dos caminhos é o de menor custo total; por fim a função $PathToGoal(T)$ retorna o caminho de menor custo total até o destino. A saída do algoritmo é uma sequência de pontos que representa o caminho planejado.

Considerando a exploração com múltiplos robôs, o algoritmo é executado separadamente para cada robô. Além disso, a posição dos robôs do sistema cooperativo é incluída no vetor de obstáculos, para evitar colisões entre eles.

Para garantir melhor desempenho na exploração do ambiente, um novo caminho é recalculado a medida que o mapa é explorado. Dessa forma, é possível garantir que o robô desloque sempre na direção do atual ponto de fronteira e evite colisão com novos obstáculos detectados.

Para minimizar o custo computacional de reconstrução das árvores, propomos um método de replanejamento que aproveita a árvore determinada anteriormente para o novo planejamento. Dessa forma, a raiz da árvore se mantém e um novo caminho é planejado até o novo destino a partir dos nós já existentes. Isso é feito eliminando todos os nós da árvore que colidem com os novos obstáculos detectados

Algoritmo 1: RRT*

Entradas:
$\boldsymbol{q_0}$ //configuração do robô (x, y, θ)
$\boldsymbol{q_f}$ //configuração final de destino (x_f, y_f, θ_f)
N //número máximo de iterações
$\mathbf{O}$ //vetor de obstáculos (x_o, y_o, r)
δ //tamanho das arestas da árvore (incremento)
T //Árvore pré-alocada ou vetor nulo
Saídas:
$\mathbf{P}$ //vetor de pontos que formam o caminho
$i = 0$;
$T \leftarrow \{q_0\}$;
while $i < N$ **do**
  $q_{rand} \leftarrow RAND()$
  $q_{near} \leftarrow NEAR(T, q_{rand})$
  $q_{new} \leftarrow STEP(q_{near}, q_{rand})$
  $Cost(q_{new}) \leftarrow$custo de q_{new} pela Eq. (3)
  if $CollisionFree(q_{near}, q_{new}, \mathbf{O})$ **then**
    $Q_{near} \leftarrow FindNear(q_{new})$
    $q_{parent} \leftarrow BestParent(q_{new}, Q_{near})$
    if $q_{parent} \neq None$ **then**
      $REWIRE(q_{parent}, Q_{near})$
      $T \leftarrow T \cup \{q_{parent}\}$
    else
      $T \leftarrow T \cup \{q_{new}\}$
    if $DistToGoal(\boldsymbol{q_f}, T)$ **then**
      $\mathbf{P} \leftarrow PathToGoal(T)$
      return $\mathbf{P}$
  $i = i + 1$;
end
return $None$

e executado o procedimento novamente até encontrar o caminho para o ponto de fronteira atual. Uma heurística é inserida para casos em que o novo caminho se afasta muito da posição atual do robô, optando-se por iniciar uma nova árvore.

O algoritmo 2 descreve essa operação de replanejamento. A função $REMOVE(\mathbf{O}, T)$ varre os nós da árvore T e remove nós e arestas que colidem com os obstáculos da matriz $\mathbf{O}$. Um fator a ser considerado é que o custo computacional aumenta a medida que o mapa é explorado, devido a inclusão de novos obstáculos e do crescimento da árvore. Se a árvore estiver com mais de 1500 nós, uma nova árvore é construída, para minimizar o custo de análise de colisões.

Algoritmo 2: Replanning

Entradas:
$\boldsymbol{q_0}$ //configuração do robô (x, y, θ)
$\boldsymbol{q_f}$ //configuração final de destino (x_f, y_f, θ_f)
N //número máximo de iterações
$\mathbf{O}$ //vetor de obstáculos (x_o, y_o, r)
δ //tamanho das arestas da árvore (incremento)
Saídas:
$\mathbf{P}$ //vetor de pontos que formam o caminho
$T \leftarrow REMOVE(\mathbf{O}, T_{old})$;
$\mathbf{P} \leftarrow RRT(\boldsymbol{q_0}, \boldsymbol{q_f}, N, \mathbf{O}, \delta, T)$;
return $\mathbf{P}$

2.4 Controle

O método de controle utilizado é baseado no cálculo do gradiente do vetor de distância da posição do robô para o caminho desejado, considerando o ponto mais próximo do caminho para o robô, que pode ser obtida da seguinte equação:

$$\nabla D = \begin{bmatrix} \frac{x - x_c}{\sqrt{(x-x_c)^2 + (y-y_c)^2}} \\ \frac{y - y_c}{\sqrt{(x-x_c)^2 + (y-y_c)^2}} \end{bmatrix}, \quad (4)$$

onde x e y são as medidas de posição do robô, e x_c e y_c os pontos da curva que forma o caminho. Dessa forma, o negativo de ∇D é um vetor que aponta para o ponto (x_c,y_c) da curva, que é o mais próximo do robô no momento. Assim, podemos determinar as velocidades necessárias para que o robô siga a curva desejada ponto a ponto a partir da seguinte equação:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = -K(\nabla D), \quad (5)$$

onde K é um fator de ganho.

O robô utilizado nos experimentos possui uma arquitetura diferencial com duas rodas e uma castor independente, como ilustra a Figura 5. Assim, utilizando o método de *feedback linearization* (Siciliano et al., 2010), é possível obter as velocidades das rodas direita (ω_r) e esquerda (ω_l) a partir de uma matriz de rotação e das velocidades V_x e V_y determinadas pelo controlador, conforme a equação a seguir:

$$\begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \cos(\theta) - \frac{L\sin(\theta)}{2d_{fb}} & \sin(\theta) + \frac{L\cos(\theta)}{2d_{fb}} \\ \cos(\theta) + \frac{L\sin(\theta)}{2d_{fb}} & \sin(\theta) - \frac{L\cos(\theta)}{2d_{fb}} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad (6)$$

onde θ é a orientação global do robô, d_{fb} é a distância escolhida para o ponto de controle (P_c), r é o raio da roda e L é a distância entre a roda e o centro de gravidade (P_o).

3. RESULTADOS

Os experimentos foram realizados em um ambiente de simulação no CoppeliaSim, conforme ilustra a Figura 1. Foram feitas diferentes execuções considerando configurações com um único robô, dois e três em um mesmo ambiente, com o objetivo de comparar o tempo gasto na exploração total do ambiente. A execução foi feita utilizando ROS e os algoritmos foram desenvolvidos em linguagem de programação Python, com exceção do Octomap que foi utilizado em sua versão de código livre [2]. Os robôs simulados são do tipo Pioneer-P3DX e equipados com sensor LiDAR Velodyne VPL-16 3D com um range limitado a 8 metros e 360° de detecção. Para fins de controle, definimos o $d_{fb} = 0.2$ m e os demais parâmetros geométricos de acordo com o robô utilizado: $L = 0.2075$ m;$r = 0.0925$ m.

No primeiro experimento com dois robôs, cada robô manteve uma velocidade média de 0.5m/s enquanto percorre os caminhos planejados. A área livre coberta por ambos os robôs foi de aproximadamente 400m^2, de um total aproximado de 890m^2 incluindo a área desconhecida. O tempo gasto total nessa exploração foi de 14, 5 minutos. A Figura 6 mostra o resultado final do mapa 3D de ocupação obtido e a Figura 7 ilustra o caminho percorrido por cada robô no processo de exploração e não os caminhos planejados. É possível observar que os pontos de destino foram selecionados de forma correta a evitar exploração repetida por ambos os robôs. Um vídeo pode ser visto em `https://youtu.be/hD6OU913Z7Q`.

[2] Octomap - `http://wiki.ros.org/octomap`

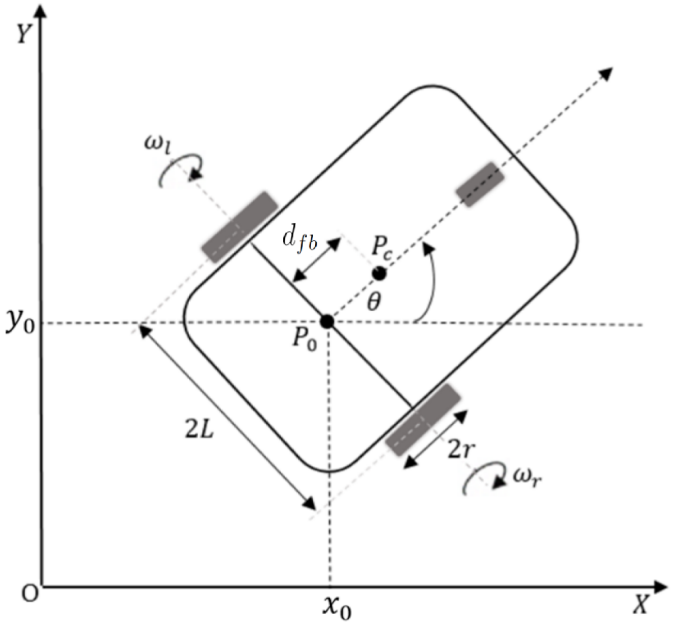

Figura 5. Modelo do robô diferencial.

Figura 6. Mapa de ocupação 3D gerado após experimento de exploração com 2 robôs.

O segundo experimento foi realizado considerando a exploração por apenas um único robô, com os mesmo parâmetros anteriores. O tempo gasto total na execução da tarefa foi de 28 minutos, quase o dobro do tempo gasto com o uso de dois robôs. Ressalta-se que com apenas um agente, a escolha do ponto de fronteira é dada pela menor distância para o robô, o que causa exploração repetida de ambientes em algumas situações. Contudo, utilizando apenas um robô foi possível mapear todo o ambiente, conforme ilustra a Figura 8.

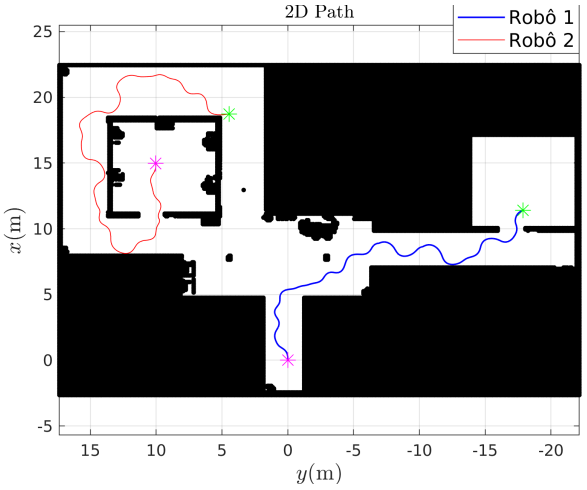

Figura 7. Caminho percorrido pelos robôs durante a exploração, partindo do marcador magenta até o ponto final em verde.

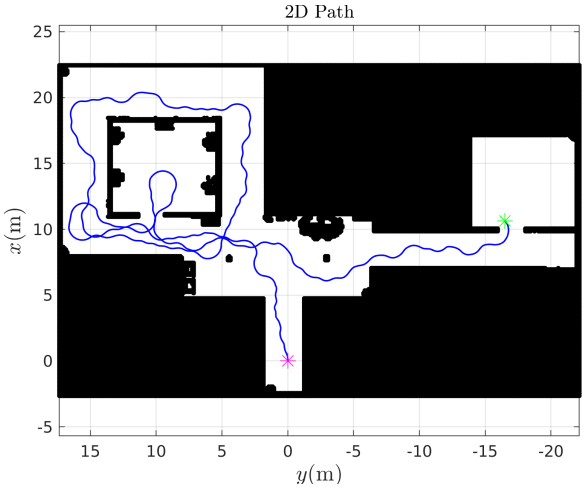

Figura 8. Caminho percorrido pelo robô durante a exploração, partindo do marcador magenta até o ponto final em verde.

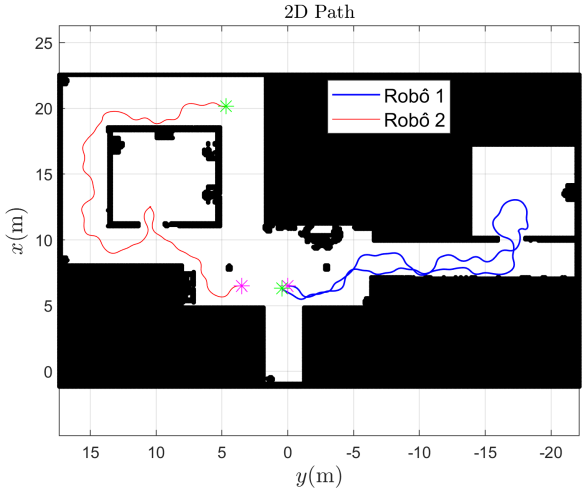

Figura 9. Caminho percorrido pelos dois robôs durante a exploração, partindo do marcador magenta até o ponto final em verde.

Um terceiro experimento foi feito para comparar a exploração utilizando dois e três robôs na mesma tarefa. A comparação é feita utilizando o mesmo ambiente anterior, porém com os robôs saindo todos de uma mesma região. Nesse caso, o experimento com dois robôs foi executado em $17,6$ minutos, enquanto o tempo gasto utilizando três robôs foi de $9,5$ minutos. As Figuras 9 e 10 apresentam

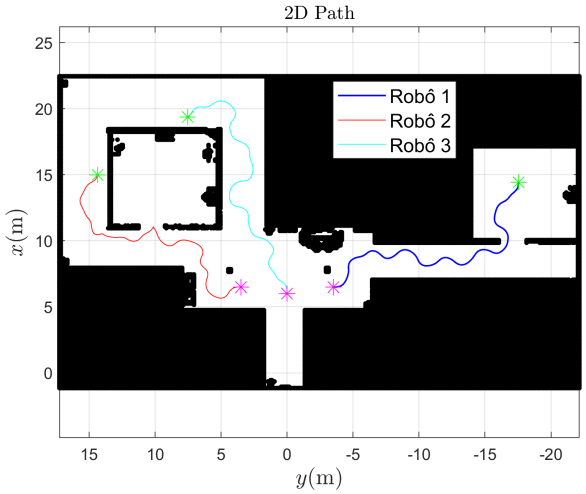

Figura 10. Caminho percorrido pelos três robôs durante a exploração, partindo do marcador magenta até o ponto final em verde.

o caminho percorrido por cada robô e o mapa gerado ao final da simulação, considerando o experimento com dois robôs e três, respectivamente. É possível observar que em um dado momento, na exploração utilizando dois robôs, o robô 1 percorre caminhos já visitados na tentativa de alcançar pontos inexplorados pelo robô 2, mas a exploração termina antes que este avance até o ponto desejado. Por fim, considerando todos os experimentos realizados, pode-se considerar que a inclusão de outros robôs móveis promove uma redução significativa no tempo total gasto para a exploração.

4. CONCLUSÕES

O trabalho apresentou um sistema de exploração autônoma por múltiplos robôs utilizando um planejador RRT* com replanejamento local. Os resultados obtidos demonstram a capacidade de exploração do método, retornando um mapa de ocupação 3D de todo o ambiente. Além disso, fica evidente que para o cenário considerado, uso de dois ou três robôs reduz o tempo total de exploração.

A definição do ponto de destino somente pela proximidade com o robô, como no caso de um único robô, se mostrou ineficaz, gerando visitações repetidas a pontos do ambiente. Apesar das heurísticas e métodos desenvolvidos na tentativa de reduzir o custo computacional no planejamento do caminho, foi possível notar que grande parte do tempo foi gasto durante o planejamento.

Como trabalho futuro pretendemos realizar experimentos em um cenário mais complexo e também com plataformas reais. Nesse sentido a inclusão de algoritmos de fusão sensorial para aprimorar a localização no ambiente será

necessária. Buscamos também, considerar um maior número de robôs e avaliar outros métodos de planejamento e controle, como técnicas de aprendizado por reforço.

Outro trabalho futuro é desenvolver novos métodos de seleção dos pontos de fronteira, com foco em aumentar a capacidade de exploração, utilizando por exemplo técnicas de aprendizado de máquina. Por fim, pretendemos incluir novos parâmetros de otimização para o planejador, como inclinações no terreno e consumo de energia.

AGRADECIMENTOS

O presente trabalho foi realizado com o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), da Fundação de Amparo à Pesquisa do Estado de Minas Gerais - Brasil (FAPEMIG), da Universidade Federal de Minas Gerais (UFMG) e do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE).

REFERÊNCIAS

Azpúrua, H., Rezende, A., Potje, G., Júnior, G.P.C., Fernandes, R., Miranda, V.R.F., et al. (2021). Towards semi-autonomous robotic inspection and mapping in confined spaces with the EspeleoRobô. *Journal of Intelligent & Robotic Systems*, 101(4), 1–27. doi:10.1007/s10846-021-01321-5.

Cieslewski, T., Kaufmann, E., and Scaramuzza, D. (2017). Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2135–2142. doi:10.1109/IROS.2017.8206030.

Colares, R.G. and Chaimowicz, L. (2016). The next frontier: combining information gain and distance cost for decentralized multi-robot exploration. In *31st Annual ACM Symposium on Applied Computing*, 268–274. doi:10.1145/2851613.2851706.

Corah, M., O'Meadhra, C., Goel, K., and Michael, N. (2019). Communication-efficient planning and mapping for multi-robot exploration in large environments. *IEEE Robotics and Automation Letters*, 4(2), 1715–1721. doi:10.1109/LRA.2019.2897368.

de Hoog, J., Cameron, S., and Visser, A. (2009). Role-based autonomous multi-robot exploration. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, 482–487. doi:10.1109/ComputationWorld.2009.14.

Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. doi:10.1007/s10514-012-9321-0.

Hu, J., Niu, H., Carrasco, J., Lennox, B., and Arvin, F. (2020). Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12), 14413–14423. doi:10.1109/TVT.2020.3034800.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894. doi:10.1177/0278364911406761.

Kulkarni, M., Dharmadhikari, M., Tranzatto, M., Zimmermann, S., Reijgwart, V., De Petris, P., Nguyen, H., Khedekar, N., Papachristos, C., Ott, L., Siegwart, R., Hutter, M., and Alexis, K. (2022). Autonomous teamed exploration of subterranean environments using legged and aerial robots. In *2022 International Conference on Robotics and Automation (ICRA)*, 3306–3313. doi:10.1109/ICRA46639.2022.9812401.

Lu, L., Redondo, C., and Campoy, P. (2020). Optimal frontier-based autonomous exploration in unconstructed environment using RGB-D sensor. *Sensors*, 20(22). doi:10.3390/s20226507.

Mir-Nasiri, N., J, H.S., and Ali, M.H. (2018). Portable autonomous window cleaning robot. *Procedia Computer Science*, 133, 197–204. doi:https://doi.org/10.1016/j.procs.2018.07.024. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

Miranda, V.R., Rezende, A., Rocha, T.L., Azpúrua, H., Pimenta, L.C., and Freitas, G.M. (2021). Autonomous navigation system for a delivery drone. *Journal of Control, Automation and Electrical Systems*, 33(1), 141–155. doi:10.1007/s40313-021-00828-4.

Niroui, F., Zhang, K., Kashino, Z., and Nejat, G. (2019). Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2), 610–617. doi:10.1109/LRA.2019.2891991.

Pathmakumar, T., Kalimuthu, M., Elara, M.R., and Ramalingam, B. (2021). An autonomous robot-aided auditing scheme for floor cleaning. *Sensors*, 21(13). doi:10.3390/s21134332.

Qin, H., Meng, Z., Meng, W., Chen, X., Sun, H., Lin, F., and Ang, M.H. (2019). Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments. *IEEE Transactions on Vehicular Technology*, 68(2), 1339–1350. doi:10.1109/TVT.2018.2890416.

Schmidt, D., Luksch, T., Wettach, J., and Berns, K. (2006). Autonomous behavior-based exploration of office environments. In *ICINCO-RA*, 235–240.

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

Sim, R. and Dudek, G. (2003). Effective exploration strategies for the construction of visual maps. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 4, 3224–3231 vol.3. doi:10.1109/IROS.2003.1249653.

Vaidis, M. and Otis, M.J.D. (2020). Toward a robot swarm protecting a group of migrants. *Intelligent Service Robotics*, 1–16. doi:10.1007/s11370-020-00315-w.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 146–151. doi:10.1109/CIRA.1997.613851.