Motion Planning of Mobile Robots in Indoor Topological Environments using Partially Observable Markov Decision Process

N. Monteiro, V. Gonçalves, and C. Maia

***Abstract*—Deterministic motion planners perform well in simulated environments, where sensors and actuators are perfect. However, these assumptions are restrictive and consequently motion planning will have poor performance if applied to real robotic systems (or a more realistic simulator), as they are inherently fraught with uncertainty. In most real robotic systems, states cannot be directly observed, and the results of the actions performed by the robots are uncertain. Thus, the robot must make use of a new class of planners that take into account system uncertainties when making a decision. In the present work, the Partially Observable Markov Decision Process is presented as an alternative to solve problems immersed in uncertainties, selecting optimal actions aiming to perform a given task. The contribution of this article is to implement the Partially Observable Markov Decision Process using greedy optimization, which has considerably simplified the decision-making problem for uncertain environments. This article also presents new ways to determine the parameters of the Partially Observable Markov Decision Process. The aforementioned tooling was applied in a system to control the actions of a real robot that navigates in a indoor topological living space with ambiguity of informations.**

***Index Terms*—Greedy optimization, Motion planning, Probabilistic robotics, Uncertainty.**

I. Introdução

Desde os primórdios, o homem tem buscado maneiras de aprimorar seus processos produtivos, tornando-os mais eficientes. Uma importante solução encontrada para tal foi a criação dos sistemas robóticos móveis. Com o avanço da tecnologia os sistemas robóticos têm se popularizado, e estão sendo desenvolvidos em grandes escalas e utilizados em uma vasta gama de situações, que abrangem desde aplicações domésticas, industriais, hospitalares às militares. Em muitas dessas aplicações, é desejado que os robôs naveguem pelo ambiente e alcancem um local predeterminado de forma autônoma. Para tal, é necessário realizar o planejamento de movimento, de modo que o robô se desloque de forma segura.

Na literatura existem várias técnicas que podem ser usadas para planejar os movimentos, cabendo ao roboticista escolher a que mais se adéqua à aplicação a ser desenvolvida. Como exemplo, podem ser citadas as técnicas que utilizam algoritmos *Bug*'s, métodos probabilísticos (tal como o RRT - *Rapidly Exploring Random Tree*) ou funções de potencial. A família de algoritmos *Bug* encontra um caminho entre duas configurações de um ambiente desconhecido. Inicialmente o robô deve prosseguir em direção ao alvo, e quando encontrar um obstáculo deve contorná-lo até poder mover na direção do alvo novamente. Este processo deve continuar até que o robô atinja o alvo [1]. O RRT é um método probabilístico, que constrói de forma incremental uma árvore de percursos viáveis, tendo como raiz a configuração inicial. Em cada iteração do RRT, um ponto aleatório (livre de colisão) é amostrado, e tenta-se conectá-lo ao nó mais próximo da árvore. O RRT se encerra quando a árvore possuir um nó na região do alvo, produzindo assim um caminho factível entre a configuração inicial e a desejada [2]. Na estratégia de funções de potencial, o movimento do robô é influenciado por um campo vetorial artificial induzido pelo alvo e pelos obstáculos presentes no ambiente. Nessa estratégia as ações de controle do robô podem ser obtidas pelo negativo do gradiente da função de potencial [1] e [3].

A maioria dos planejadores de movimentos clássicos, como os citados acima, consideram que as ações de controle executadas por um robô são determinísticas, e que os sensores são capazes de mensurar completamente os estados do robô. No entanto, estas suposições são restritivas, pois os sistemas reais são inerentemente carregados de incertezas [4] e [5].

Uma alternativa aos métodos clássicos é o PDMPO (Processo de Decisão de Markov Parcialmente Observável), que fornece um arcabouço para resolver problemas de tomada de decisão estocásticos. No PDMPO há possibilidade de um agente intervir no sistema tomando decisões (executando ações), e cada decisão escolhida possui um resultado incerto. Para cada ação é atribuída uma recompensa, que depende do estado que o sistema se encontra. Uma particularidade do PDMPO é que os estados do sistema não podem ser medidos diretamente, sendo inferidos por meio de observações indiretas do ambiente. Na resolução do PDMPO busca-se encontrar uma política (mapeamento da crença de estados em ações) ótima, que determina qual decisão tomar para maximizar a recompensa esperada de uma sequência de decisões [4], [6] e [7]. As grandes vantagens do PDMPO, em relação aos planejadores de movimento clássicos, são: integrar a etapa de localização no ambiente (estimação de estados por meio das percepções) e a de tomada de decisão em uma única estrutura; e considerar as incertezas presentes nessas duas etapas.

Algumas pesquisas utilizando o PDMPO no contexto de planejamento de caminhos de robôs, em ambientes topológicos internos, são mostradas a seguir. Koenig e Simmons [8] usaram

N. S. Monteiro. Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, neemias@ufmg.br.

V. M. Gonçalves. Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, mariano@cpdee.ufmg.br.

C. A. Maia. Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, maia@cpdee.ufmg.br.

o PDMPO como arquitetura de navegação para o Xavier, robô destinado a entrega de objetos em um ambiente de escritórios. O Xavier pode ser monitorado e controlado por uma interface de internet. Nesta aplicação, o ambiente é constituído por salas e corredores, e é abstraído por meio de um mapa topológico, no qual os estados são definidos com espaçamento de 1m de distância. No artigo de Páll et al. [9], o PDMPO é utilizado para um problema de assistência doméstica em um ambiente de trabalho discretizado. Um robô é responsável por monitorar os estados parcialmente observáveis de interruptores, e os desliga se necessário. Na pesquisa de Wang et al. [10], o PDMPO é usado para realizar buscas de objetos em ambientes que possuem a estrutura topológica. No planejamento do caminho há uma ponderação entre o custo do caminho a ser executado e o ganho de informações no processo de busca. Outra aplicação de localização de objetos foi proposta por Wandzel et al. [11], sendo esta motivada pela situação de encontrar sobreviventes humanos em um local de desastre.

Neste artigo, é proposto empregar o PDMPO no planejamento da tomada de decisão de robôs móveis terrestres, que se movem em ambientes topológicos internos ambíguos. Nestes locais, a tomada de decisão se torna mais difícil devido à indistinguibilidade entre várias partes do ambiente, e a falta de acesso às medições de GPS [12]. Para suprir a falta de GPS, a estimativa da localização do robô móvel será realizada usando marcos visuais (*landmarks*) distribuídos pelo ambiente. A partir da estimativa de localização, o PDMPO determina políticas de controle ótimas a serem executadas pelo robô com o propósito de alcançar um local predeterminado do ambiente.

Os métodos tradicionais de resolução do PDMPO são complexos, exigem alto custo computacional, e têm aplicações limitadas a problemas pequenos [13] e [14]. Uma alternativa para resolver o PDMPO, apresentada por Pineau e Gordon [15], é o PEMA (*Point-based Error Minimization Algorithm*). Este algoritmo consiste em delimitar as crenças em poucos pontos de operação, resolvendo o problema com maior rapidez que os métodos descritos em Aberdeen [13] e Braziunas [14]. Entretanto, a grande dificuldade do PEMA está em selecionar os pontos de crença candidatos que irão minimizar o erro do PDMPO. Algumas abordagens utilizam aproximações e consideram que o PDMPO é completamente observável, como mostram Koenig e Simmons [8]. Outras realizam abstrações das ações e dos estados, agrupando-os em blocos, como propõem Kostavelis et al. [16]. No entanto, essas abordagens geralmente levam a resultados muito abaixo do ideal.

Diante da intratabilidade e dificuldade dos métodos descritos acima, a primeira contribuição do artigo é implementar o PDMPO através da otimização gananciosa, resultando em um algoritmo de seleção de políticas de controle com com baixo esforço computacional, possuindo complexidade $\mathcal{O}(|A|\,|O|\,|S|^2)$, sendo: A, O e S o conjunto de ações, observações e estados, respectivamente. Na aplicação proposta, o PDMPO será responsável pelo planejamento das ações em alto nível, determinado os estados que devem ser alcançados; e o planejador local apresentado no Apêndice A será responsável pelas ações de baixo nível, ou seja, as ações que o robô deve executar para navegar de um estado para outro. O planejador local cria trajetórias de locomoção entre estados por meio de funções polinomiais.

A segunda contribuição do artigo é propor uma função de recompensa, dada em função do estados e das ações, que induz o robô a chegar no estado do sistema marcado como alvo.

O artigo está organizado da seguinte forma: na Seção II, é feita uma explanação sobre o PDMPO e as modificações introduzidas. Na Seção III, é descrito o aparato experimental utilizado neste trabalho. Na Seção IV, são mostrados os resultados da aplicação do PDMPO no controle das ações de um robô real, que se locomove em um espaço de convivência topológico com ambiguidade de informações. Na Seção V, são apresentadas as conclusões e sugestões de trabalhos futuros.

II. METODOLOGIA: PDMPO

O problema da navegação de robôs móveis em ambientes topológicos (a representação topológica consiste em abstrair o espaço de trabalho em grafos formados por estados e arestas; os estados simbolizam locais de interesse e as arestas as conexões entre estes locais [17]) incertos pode ser resolvido com o Processo de Decisão de Markov Parcialmente Observável. O PDMPO é um método probabilístico que seleciona uma sequência de ações com o objetivo de realizar uma tarefa, dadas as imprecisões do sistema [7]. O termo parcialmente observável indica que as medições do sensores são incompletas e/ou projeções ruidosas dos estados do sistema [4] e [18]. Neste artigo, considera-se que os estados, as ações, as observações e o tempo são discretos.

A. Conceitos

O PDMPO é constituído por: ações $a \in A$ que podem ser executadas no estados $s \in S$ do sistema; função de recompensa $R : S \times A \mapsto \Re$, que indica o ganho que o sistema terá dado um estado s e uma ação a; função de transição de estados $\Gamma : S \times A \times S \mapsto [0, 1]$ denotada pela probabilidade $p(s'|a, s)$; observações $o \in O$ indiretas (que são resultado da execução de ações no ambiente) dos estados $s \in S$; e função de observabilidade $\Omega : S \times O \mapsto [0, 1]$ denotada pela probabilidade $p(o|s)$ [6].

Em vez de um "estado atual" do sistema, no PDMPO há uma distribuição de probabilidade sobre os estados, também chamada de crença Bel, sendo $Bel(s)$ a probabilidade do sistema estar no estado s, e $\sum_s Bel(s) = 1$. Em um sistema parcialmente observável, a crença $Bel(s)$ deve ser atualizada a cada tomada de decisão por meio das observações o provenientes do sistema. Neste trabalho, a crença $Bel(s)$ será representada por meio do procedimento ES-MOM (Entrada Saída - Modelo Oculto de Markov), apresentado com mais detalhes na Seção II-C. Alguns termos presentes no contexto dos PDMPO's são [4], [6] e [7]:

- *Horizonte de planejamento (t):* é o número de épocas de decisão (t) disponível para se tomar decisões. Ele pode ser divido em três casos: $t = 1$, também conhecido como ganancioso; t finito, porém maior que 1; e t infinito.
- *Política (π):* é o mapeamento da crença de estados $Bel(s)$ em uma ação a, e é definida pela função $\pi : \Upsilon \mapsto A$, no qual, $\Upsilon = \left\{Bel \in \Re^{|S|} | \sum_s Bel(s) = 1, Bel(s) \geq 0\right\}$ é o espaço de todas as crenças $Bel(s)$.

- *Função valor (V):* mensura o valor esperado de uma política específica, e é dada pela função $V : \Upsilon \mapsto \Re$.

Segundo Thrun et al. [4], a política π_t e a função valor V_t para um horizonte t, $0 \leq t \leq T$, são dadas por (1) e (2), respectivamente. Sendo (2) conhecida como Equação de Bellman Estocástica.

$$\pi_t(Bel) = \underset{a}{\operatorname{argmax}} \left[\sum_{Bel'} [R_B(Bel', a) + V_{t-1}(Bel')] \cdot p(Bel'|a, Bel) \right], \quad (1)$$

$$V_t(Bel) = \max_a \left[\sum_{Bel'} [R_B(Bel', a) + V_{t-1}(Bel')] \cdot p(Bel'|a, Bel) \right], \quad (2)$$

no qual,

$$R_B(Bel', a) = \sum_{s'} R(s', a) Bel'(s'). \quad (3)$$

O cálculo da função valor pode ser obtido de maneira relativamente simples no espaço de estados [19]. Entretanto, o cálculo da função valor no espaço de crenças (como é o caso de (2)) é altamente complexo. A função valor V_t é uma função sobre a distribuição de probabilidade Bel. Então, se o espaço de estados S é finito, o espaço de crenças será contínuo e de dimensão $S - 1$, tornando o cálculo de V_t uma tarefa muito onerosa [4]. O problema de encontrar uma função valor ótima V_t para um PDMPO com horizonte finito exige alto custo computacional [20], e para horizonte infinito é indecidível, isto é, não existe garantia que a função valor será ótima [21].

B. Otimização Gananciosa

Diante da intratabilidade dos métodos clássicos, este artigo resolve o PDMPO usando a otimização gananciosa (horizonte $t = 1$). Deste modo, será calculada uma política e uma função valor, para cada época de decisão, independente das decisões tomadas anteriormente. Assim, (1) e (2) podem ser reduzidas:

$$\pi(Bel) = \underset{a}{\operatorname{argmax}} \left[\sum_{Bel'} R_B(Bel', a) p(Bel'|a, Bel) \right], \quad (4)$$

$$V(Bel) = \max_a \left[\sum_{Bel'} R_B(Bel', a) p(Bel'|a, Bel) \right]. \quad (5)$$

A probabilidade $p(Bel'|a, Bel)$ é uma distribuição sobre a crença Bel' dada uma crença Bel e uma ação a. Se apenas Bel e a são conhecidas, Bel' não será única e $p(Bel'|a, Bel)$ será uma distribuição sobre todas as crenças. Entretanto, se a observação o' (após a execução de a) é conhecida, a crença posterior Bel' será única e $p(Bel'|a, Bel)$ passa a ser uma distribuição discreta degenerada [4], dada por:

$$p(Bel'|a, Bel) = \sum_{o'} p(Bel'|a, Bel, o') p(o'|a, Bel), \quad (6)$$

em que, $p(Bel'|a, Bel, o')$ assumirá valor 1 apenas se a ação a, tomada na crença de estados Bel, obtendo a observação o', levar à crença de estados Bel'. Aplicando (6) em (5), tem-se:

$$V(Bel) = \max_a \left[\sum_{o'} \underbrace{\left[\sum_{Bel'} R_B(Bel', a) p(Bel'|a, Bel, o') \right]}_{(\star)} \cdot p(o'|a, Bel) \right]. \quad (7)$$

O somatório $(\star)$ terá apenas um termo diferente de zero, e então se tornará obsoleto. Quando isso acontece a crença, é:

$$Bel'(s') = \frac{1}{p(o'|a, Bel)} p(o'|s') \sum_s p(s'|a, s) Bel(s). \quad (8)$$

A dedução de (8) pode ser vista com detalhas em Thrun et al. [4]. Com posse de (8), a expressão (7) é reescrita:

$$V(Bel) = \max_a \left[\sum_{o'} R_B(Bel', a) p(o'|a, Bel) \right]. \quad (9)$$

De acordo com (3), a recompensa R_B em função de (8) é:

$$R_B(Bel', a) = \sum_{s'} R(s', a) \frac{1}{p(o'|a, Bel)} p(o'|s') \cdot \sum_s p(s'|a, s) Bel(s). \quad (10)$$

Então, substituindo (10) em (9):

$$V(Bel) = \max_a \left[\sum_{o'} \sum_{s'} R(s', a) p(o'|s') \cdot \sum_s p(s'|a, s) Bel(s) \right]. \quad (11)$$

Comparando (5) e (11), pode ser notado que o somatório que antes era realizado no espaço de crenças passou a ser computado de maneira mais simples no espaço de estados e observações. A maximização pode ser dividida sobre as ações:

$$V_t(Bel_t, a) = \sum_{o'} \sum_{s'} R(s', a) p(o'|s') \sum_s p(s'|a, s) Bel_t(s), \quad (12)$$

$$V_t^*(Bel_t) = \max_a V_t(Bel_t, a). \quad (13)$$

Em (12) e (13) foi incluída a notação t referindo-se à época de decisão. O termo $V_t(Bel_t, a)$ é a função valor sobre a crença Bel_t, assumindo que a próxima ação será a. Usando $V_t(Bel_t, a)$, a política ótima associada pode ser determinada:

$$\pi_t^*(Bel_t) = \underset{a}{\operatorname{argmax}} V_t(Bel_t, a). \quad (14)$$

C. Estimação da Crença: ES-MOM

Nos sistemas reais, o conjunto de estados S raramente é observável. Em um sistema parcialmente observável, a crença $Bel(s)$ deve ser atualizada a cada tomada de decisão utilizando as observações provenientes do sistema. Neste trabalho, a crença $Bel(s)$ será representada por meio do procedimento ES-MOM (Entrada Saída - Modelo Oculto de Markov) [22]. O ES-MOM é um método de estimação de estados probabilístico, no qual os estados não são diretamente observáveis, ou seja, ocultos. O ES-MOM representa o cenário em que a observação o é uma função probabilística do verdadeiro estado do sistema, e será usado para localizar o estado mais provável de um robô

móvel em um ambiente topológico. Usando o ES-MOM, a crença de estados pode ser inferida de forma iterativa [23]:

$$Bel_t(s') = p(o_t|s') \left[\sum_s p(s'|a_{t-1}, s) Bel_{t-1}(s) \right], \forall\, s' \in S. \tag{15}$$

D. Funcionamento do PDMPO

A Fig. 1 apresenta um esquemático da dinâmica de funcionamento do PDMPO usado no presente trabalho. Em cada época de decisão, o PDMPO utiliza as informações da última ação executada a, da última observação coletada o e da última crença sobre os estados, para determinar a crença atual $Bel_t(s)$. Com base em (14) e na crença atualizada, uma nova política de controle π (ação) é recomendada. O efeito da execução da política no ambiente é retornado por meio de uma nova observação o, que posteriormente será utilizada no ES-MOM. Este processo se repete iterativamente [6].

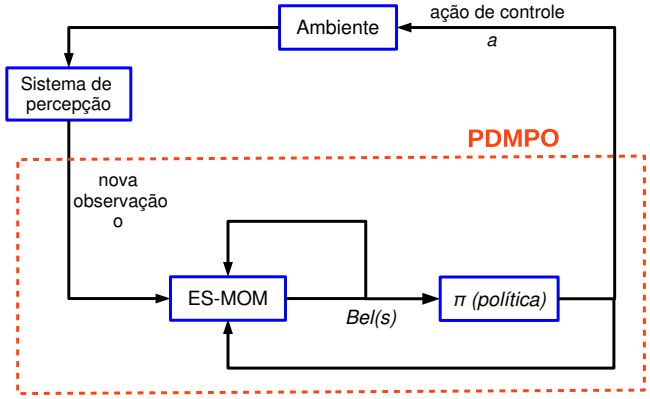


Fig. 1. Esquemático de funcionamento do PDMPO.

Utilizando (12), (13) e (14), o Algoritmo 1 é proposto para resolver o PDMPO de forma iterativa usando a otimização gananciosa. O Algoritmo 1 recebe como entrada o PDMPO $(S, A, \Gamma, R, \Omega, O)$, o estado marcado como alvo s_{goal}, e a crença inicial sobre os estados. Ele fornece como saída a política $\pi^*(Bel)$ a ser executada e a função valor $V^*(Bel)$ associada. Da linha 8 à 12, é verificada a possibilidade do sistema passar para o estado s' quando a ação a é executada; da linha 6 à 14 é calculada a probabilidade de se observar o' no estado s' vezes a recompensa do estado s'. Este processo é repetido $\forall\, s' \in S$. Entre a linha 2 à 18, é realizado o processo acima $\forall\, a \in A$ e $\forall\, o' \in O$. Na linha 19, é computada a política π^* que maximiza a recompensa esperada imediata V^* (linha 20). Após implementar a política π^* (linha 21), usando o planejador de movimento apresentado no Apêndice A, o robô deve atualizar a crença de estados Bel com o ES-MOM (linha 22). Caso $Bel(s_{goal}) > 0.7$, é considerado que o robô chegou ao alvo, e o algoritmo encerrará retornando *sucesso* (linha 24). Se o estado alcançado não for o alvo ($Bel(s_{goal}) \leq 0.7$), o algoritmo retorna à linha 1 e repete o processo descrito acima até alcançar o estado desejado s_{goal} [5].

Algoritmo 1: PDMPO usando otimização gananciosa

Entrada: PDMPO $(S, A, \Gamma, R, \Omega, O)$
Estado alvo s_{goal}
Crença inicial sobre os estados Bel_0
Saída: Política π^* e função valor V^* (em cada época de decisão)

```
1  início
2      para todo a ∈ A faça
3          h ← 0
4          para todo o' ∈ O faça
5              g ← 0
6              para todo s' ∈ S faça
7                  f ← 0
8                  para todo s ∈ S faça
9                      se a ∈ A_s então
10                         f ← f + p(s'|a, s)Bel(s)
11                     fim
12                 fim
13                 g ← g + R(s', a)p(o'|s')f
14             fim
15             h ← h + g
16         fim
17         V(Bel, a) ← h
18     fim
19     π*(Bel) ← argmax_a V(Bel, a)
20     V*(Bel) ← max_a V(Bel, a)
21     executar π*
22     executar ES-MOM (atualizar Bel)
23     se robô chegou ao estado alvo então
24         retorna sucesso
25     fim
26     senão
27         retorna à linha 1
28     fim
29 fim
```

E. Análise da Complexidade Computacional

Para avaliar o comportamento do Algoritmo 1, em termos de recursos computacionais, é realizado o estudo da complexidade computacional. A análise de complexidade traduz o comportamento do algoritmo em função dos parâmetros de entrada do PDMPO. A complexidade do Algoritmo 1 pode ser estabelecida em termos das linhas 2 à 18, e das linhas 19 e 20. A complexidade referente aos quatro "*for's*" presentes nas linhas 2 à 18 é $\mathcal{O}(|A|\,|O|\,|S|^2)$. Nas linhas 19 e 20 são realizadas operações de comparação (max e argmax) que dependem da cardinalidade de A, resultando em complexidade $\mathcal{O}(|2A|)$. Assim, a complexidade total é $\mathcal{O}(|A|\,|O|\,|S|^2 + |2A|) = \mathcal{O}(|A|\,(|O|\,|S|^2 + 2))$. Quando o tamanho de O e S crescem muito o 2 se torna irrelevante, então realizando a simplificação assintótica da constante aditiva, o termo que que domina a complexidade do Algoritmo 1 é $\mathcal{O}(|A|\,|O|\,|S|^2)$. A seguir, é apresentada a complexidade computacional de algoritmos de resolução do PDMPO, disponíveis na literatura, usados para *benchmark* do algoritmo proposto.

Um dos principais algoritmos utilizados para resolver o PDMPO é a iteração de valores. Ele tem complexidade $\mathcal{O}\left(|A|N^{|S|-1}\right)$, que cresce exponencialmente em função S, sendo: N o tamanho da discretização da crença de estados [6]. O PBVI é uma aproximação do algoritmo de iteração de valores e apresenta complexidade $\mathcal{O}\left(|S|\,|A|\,|V_{t-1}|\,|O|\,|B|\right)$, sendo: $|B|$ o tamanho de um conjunto de pontos crenças representativas e $|V_{t-1}|$ o número de componentes lineares requeridos para representar a função valor em $t-1$ [24].

Em Foka e Trahanias [25] é utilizado o PDMPO hierárquico, cujo objetivo é "quebrar" o PDMPO original em vários

PDMPO's relacionados, que possuem um pequeno subconjunto de ações e estados. Na divisão hierárquica, os PDMPO's de nível superior têm controle direto sobre os PDMPO's de nível inferior. A política para o PDMPO de alto nível consiste na seleção de submódulos apropriados, por outro lado, as políticas de baixo nível compreendem ações diretas no domínio do problema. A ordem de complexidade deste algoritmo é $\mathcal{O}\left(\left(\frac{|S|}{2^{2(l-1)}}\right)^2 |V_{t-1}|^{|O|}\right)$, sendo: l a quantidade de níveis hierárquicos (geralmente são valores pequenos).

A complexidade dos algoritmos desta subseção foi calculada para o instante de tempo t. A iteração de valores apresenta custo elevado, pois depende exponencialmente de $|S|$, além de ser sensível a discretização da crença. O PDMPO hierárquico tem o problema da formulação adequada do níveis l e apresenta complexidade dada, exponencialmente, em função de $|O|$. O PBVI tem a dificuldade da escolha de pontos de crenças que sejam realmente representativos e $|V_{t-1}|$ pode crescer com o aumento de t. Desta forma, o algoritmo proposto possui menor ordem de complexidade do que os supracitados, pois depende linearmente de $|A|$ e $|O|$, e quadraticamente de $|S|$.

F. Parâmetros do PDMPO

A seguir serão apresentados os métodos propostos para calcular os parâmetros $p(s'|a,s)$, $R(s',a)$ e $p(o'|s')$ do PDMPO.

1) Probabilidade de transição de estados: O termo $p(s'|a,s)$ indica a probabilidade do sistema passar para o estado s', assumindo que ele estava no estado s, e foi executada a ação a. A ação discreta a é uma representação em alto nível, e indica a sequência de movimentos que devem ser executados para deslocar o robô de um estado para outro. Esta ação engloba uma série de ações contínuas $\mathbb{U} = (u_1, ..., u_{|\mathbb{U}|})$. Portanto, tem-se $p(s'|a,s) = p(s'|\mathbb{U},s)$. No Apêndice A é mostrado o planejador local desenvolvido para gerar a sequência de movimentos $\mathbb{U}$.

Executando a sequência de ações $\mathbb{U}_t$ em um estado arbitrário s, e utilizando o estimador probabilístico de estados contínuos EKF (*Extended Kalman Filter*) [4] e [26], é possível predizer em qual local do mapa topológico o robô estará no instante de tempo $t+1$. Este local é simbolizado por $\bar{s}$ e será um estado "virtual" no mapa, ou seja, um artifício para estimar a probabilidade de transição, como ilustra a Fig. 2.

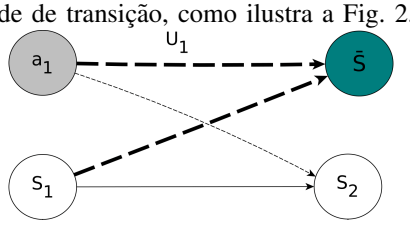


Fig. 2. Modelo gráfico incorporando o estado virtual $\bar{s}$, usado para calcular a probabilidade de transição $p(s'|\mathbb{U},s)$, exemplificada por $p(s_2|\mathbb{U}_1,s_1)$.

Assim, a probabilidade de transição pode ser obtida por:

$$p(s'|a,s) = p(s'|\mathbb{U},s) = \frac{f(\bar{s},s')}{\sum_{s'} f(\bar{s},s')}, \tag{16}$$

no qual, $f(\bar{s},s')$ é um função de densidade de probabilidade (PDF), isto é, uma distribuição gaussiana centrada em $\bar{s}$:

$$f(\bar{s},s') = e^{-\frac{1}{2}\left(\frac{(s'_x-\bar{s}_x)^2}{\sigma^2}+\frac{\left(s'_y-\bar{s}_y\right)^2}{\sigma^2}\right)}. \tag{17}$$

O somatório no denominador de (16) é necessário para normalizar a probabilidade de transição. Os estados (s e s') são posições $(x,y)^T$ no mapa, assim, é adicionada uma orientação "fictícia" ao estado s para aplicar o EKF e calcular $\bar{s}$. Esta orientação será a que o robô possuía antes de executar $\mathbb{U}$. Como a probabilidade de transição é dada em função de $\mathbb{U}$, ela deverá ser calculada em cada iteração do PDMPO. A variável σ indica o desvio padrão da distribuição gaussiana $f(\bar{s},s')$.

2) Recompensa: A função $R(s',a)$ incentiva o PDMPO a seguir o caminho com menor distância até o alvo, e retrata a recompensa do estado s' quando a ação a é executada. Dado $a \therefore a_{s_i,s_j}$, uma ação factível que leva o robô do estado s_i para o estado s_j, a recompensa $R(s',a)$ será calculada por:

$$R(s',a) = \begin{cases} \frac{1}{1+D_{s'}+d_{s_i,s_j}}, & \text{se } s_j = s' \\ 0, & \text{se } s_j \neq s' \end{cases}. \tag{18}$$

A variável $D_{s'}$ representa a distância ótima do estado s' ao estado alvo, e é obtida pela implementação do algoritmo de Dijkstra (usando como critério de otimalidade a distância euclidiana). A variável d_{s_i,s_j} representa a distância entre os estados s_i e s_j, e mensura o custo obtido ao se executar a_{s_i,s_j}.

A expressão (18) foi a maneira proposta neste artigo para determinar a função de recompensa $R(s',a)$, e por construção a mesma induz o robô chegar no alvo. Quanto mais distante o estado s' se encontrar do alvo, maior será o termo '$1 + D_{s'} + d_{s_i,s_j}$' e consequentemente, a recompensa $R(s',a)$ será menor (valores próximos de zero para estados s' distantes). Portanto, as ações a que levam o robô a estados s' distantes do alvo serão penalizadas com baixa recompensa. Por outro lado, ações que levam o robô a se aproximar do alvo apresentam termos '$1+D_{s'}+d_{s_i,s_j}$' com amplitudes menores, e consequentemente maiores recompensas $R(s',a)$. Assim, (18) incentiva o robô a seguir o caminho com menor distância (topológica), à medida que fornece recompensas maiores para ações a que o direcionem ao estado alvo.

3) Probabilidade de observabilidade: As *landmarks* (marcos visuais) são informações fortemente distinguíveis no ambiente, e no contexto da robótica são usadas para fins de localização. Se um robô encontrar uma *landmark*, e esta aparece em um mapa, consequentemente o robô é localizado em relação ao mapa. Elas são colocadas em locais fáceis de serem identificadas, e são rotuladas com uma assinatura (ID). A *landmark* detectada pelos sensores do robô, em um dado instante, constitui uma observação indireta o' (o_t para o caso de (15)) do verdadeiro estado [4]. Assim, $p(o'|s')$ mensura a chance da observação o' ser percebida no estado s'.

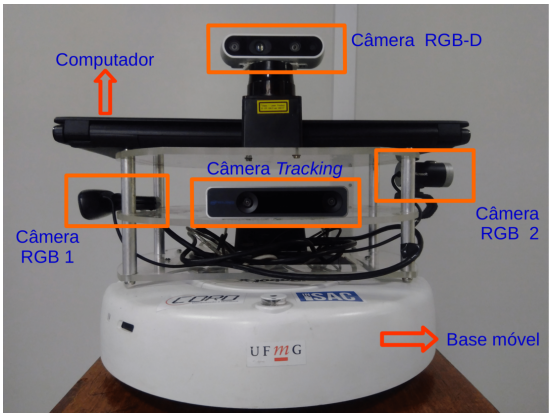
Neste artigo, é considerado que o mapa topológico do ambiente é fornecido a *priori*, sendo conhecidos os estados que uma dada *landmark* pode ser observada. Portanto, se a observação o' pode ser vista no estado s', tem-se $p(o'|s') = 0.85$. Para as demais observações $o \in O \backslash o'$, tem-se $p(o|s') = 0.15/(|O|-1)$. Obtendo assim, $\sum_o p(o|s') = 1, \forall\, s' \in S$.

III. Plataforma Robótica e Landmark

Nos experimentos deste artigo foi utilizada a plataforma robótica mostrada na Fig. 3a. Ela é composta por uma base móvel não holonômica iRobot Create, um computador Intel

Core i7-7500U com sistema Ubuntu 16.04, duas câmeras monoculares RGB, uma câmera RGB-D Intel RealSense D435 (não foi usada nos experimentos) e uma câmera Intel RealSense *Tracking* T265 (cedida pelo ITV). O *framework* usado para integrar o *hardware* e o *software* foi o ROS (*Robot Operating System*). O processamento das imagens capturadas pelas câmeras foi realizado por meio da biblioteca computacional OpenCV e os programas desenvolvidos em linguagem C++.

A câmera *Tracking* T265 fornece o rastreamento da pose do robô por meio de um tópico do ROS, e esta medição é usada na parte de atualização do EKF (mencionado na Seção II-F). As câmeras RGB's (colocadas nas laterais do robô) são usadas para identificar *landmarks*. A *landmark* usada nos experimentos foi o Aruco (Fig. 3b), um marcador robusto a variações de luminosidade e pontos de vista; que possui uma borda preta e uma matriz binária interna que determina sua ID. Para identificar os Arucos foi usada a ArUco, biblioteca *open source* desenvolvida por Garrido-Jurado et al. [27].



(a) Plataforma robótica. (b) Aruco.

Fig. 3. Robô utilizado nos experimentos e o marcador Aruco [27].

IV. Resultados

A seguir, são apresentados os resultados práticos do artigo. Na primeira parte é discutido o experimento do ES-MOM e na segunda o experimento do PDMPO.

A. Experimento: ES-MOM

Nesta seção, é mostrado um experimento realizado com o ES-MOM no mapa topológico da Fig. 4. Este mapa é uma abstração topológica do 2º andar da Escola de Engenharia da UFMG. O mapa possui 39 estados, e próximo a cada estado existe um Aruco. Os Arucos possuem uma ID que varia de 1 a 8, de acordo com a relação apresentada na Tabela I. As linhas azuis mostram as conexões factíveis entre os estados.

Pode ser notado, pela Fig. 4 e Tabela I, que este ambiente é simétrico e há ambiguidade ao relacionar uma observação a um estado. Por exemplo, a ID 1 pode ser vista nos estados s_1, s_7, s_{15} e s_{21}; e quando o robô detectá-la não conseguirá distinguir em qual estado está. Esta indistinguibilidade se torna uma dificuldade, pois há muita ambiguidade ao se relacionar uma observação a um estado do sistema. É assumido que o robô começa o experimento em um estado (mas ele não sabe qual) e que inicialmente está orientado de acordo com o sistema de referência inercial do mapa. Dadas estas

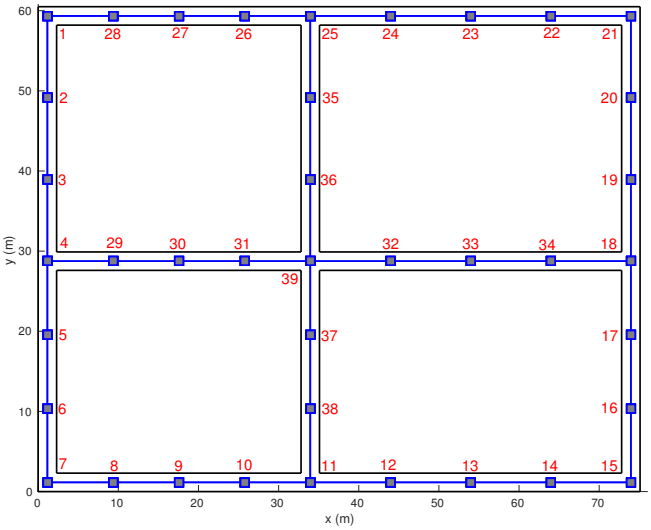


Fig. 4. Mapa topológico (em escala real) de um andar de um prédio.

TABELA I
OBSERVAÇÕES (ID'S) REFERENTES A CADA ESTADO (ES-MOM).

ID	Estados
1	$s_1 - s_7 - s_{15} - s_{21}$
2	$s_4 - s_{11} - s_{18} - s_{25}$
3	$s_2 - s_6 - s_{16} - s_{20} - s_{35} - s_{38}$
4	$s_3 - s_5 - s_{17} - s_{19} - s_{36} - s_{37}$
5	$s_8 - s_{14} - s_{22} - s_{28} - s_{29} - s_{34}$
6	$s_9 - s_{13} - s_{23} - s_{27} - s_{30} - s_{33}$
7	$s_{10} - s_{12} - s_{24} - s_{26} - s_{31} - s_{32}$
8	s_{39}

informações, o ES-MOM foi implementado para estimar a localização do robô (Fig. 3a), enquanto ele percorria o trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$ (do mapa da Fig. 4).

O resultado é mostrado na Tabela II, cuja primeira coluna apresenta o instante de tempo t que a crença é tomada, e a segunda coluna o verdadeiro estado que o robô estava no respectivo t. Nas demais colunas são mostrados os quatro maiores valores de probabilidade $Bel(s')$ em cada instante de tempo (em ordem decrescente). Apesar do mapa da Fig. 4 ser complexo e da falta de conhecimento prévio sobre a configuração inicial do robô, o ES-MOM conseguiu convergir para localização real (com boa margem de confiança) na sexta iteração, e mostrou ser um método de estimação de localização robusto. Um vídeo ilustrativo de suporte a este experimento pode ser visualizado em https://youtu.be/7n5mvFTyJi8.

TABELA II
PROBABILIDADE $Bel(s')$ DO ES-MOM PARA O TRAJETO $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$.

t	Robô	$Bel(s')$	$Bel(s')$	$Bel(s')$	$Bel(s')$
1	s_7	s_1= 0.21	s_7= 0.21	s_{15}= 0.21	s_{21}= 0.21
2	s_8	s_8= 0.30	s_{28}= 0.30	s_{29}= 0.30	s_{14}= 0.02
3	s_9	s_9= 0.32	s_{27}= 0.32	s_{30}= 0.32	s_{13}= 0.01
4	s_{10}	s_{10}= 0.33	s_{26}= 0.33	s_{31}= 0.33	s_{12}= 0.00
5	s_{11}	s_{11}= 0.48	s_{25}= 0.48	s_{31}= 0.00	s_{10}= 0.00
6	s_{38}	s_{38}= 0.94	s_{35}= 0.06	s_{11}= 0.00	s_{10}= 0.00
7	s_{37}	s_{37}= 0.98	s_{36}= 0.02	s_{39}= 0.00	s_{38}= 0.00
8	s_{39}	s_{39}= 0.99	s_{38}= 0.00	s_{37}= 0.00	s_{36}= 0.00
9	s_{31}	s_{31}= 0.99	s_{32}= 0.00	s_{39}= 0.00	s_{38}= 0.00
10	s_{30}	s_{30}= 0.99	s_{39}= 0.00	s_{38}= 0.00	s_{37}= 0.00
11	s_{29}	s_{29}= 0.99	s_{31}= 0.00	s_{39}= 0.00	s_{38}= 0.00
12	s_4	s_4= 0.99	s_{18}= 0.00	s_{39}= 0.00	s_{38}= 0.00
13	s_5	s_5= 0.95	s_3= 0.05	s_{39}= 0.00	s_{38}= 0.00
14	s_6	s_6= 0.99	s_2= 0.00	s_4= 0.00	s_{39}= 0.00

B. Experimento: PDMPO

Nesta seção, são apresentados experimentos reais usando o Algoritmo 1 para resolver o PDMPO, selecionando ações de controle ótimas de forma autônoma. O cenário escolhido foi um espaço de convivência próximo à biblioteca da Escola de Engenharia da UFMG. Uma perspectiva do ambiente é mostrada na Fig. 5. Este ambiente é constituído basicamente de sofás e *poofs*, possui uma dimensão de 22.4m x 8.3m, tem 21 estados e 58 ações factíveis. O espaço pode ser divido em três áreas, conforme detalha a Fig. 6.

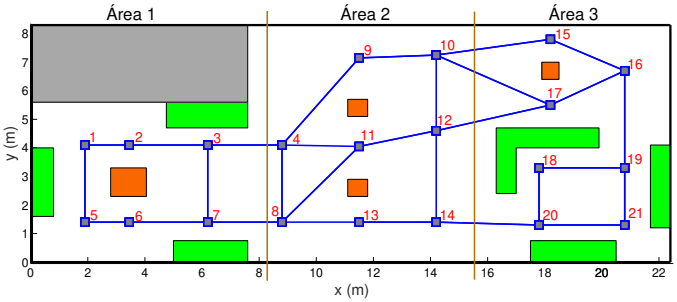


Fig. 5. Perspectiva do espaço de convivência utilizado nos experimentos do PDMPO. Os retângulos verdes simbolizam sofás, os laranjas *poofs*, e o cinza uma escada. As linhas azuis mostram as conexões possíveis entre os estados.

(a) Imagem da Área 1.

(b) Imagem da Área 2.

(c) Imagem da Área 3.

Fig. 6. Espaço de convivência usado nos experimentos do PDMPO.

Próximo a cada estado do ambiente existe um Aruco (usados como *landmark*) possuindo uma ID que varia 1 a 10, conforme a relação apresentada na Tabela III. Este ambiente também é ambíguo, pois uma ID pode ser vista em mais de um estado, incluindo assim incerteza na etapa de estimação de estados.

TABELA III
OBSERVAÇÕES (ID'S) REFERENTES A CADA ESTADO (PDMPO).

ID	Estados
1	$s_1 - s_5$
2	$s_2 - s_6$
3	$s_3 - s_7$
4	$s_4 - s_8$
5	$s_9 - s_{11} - s_{13}$
6	$s_{10} - s_{12} - s_{14}$
7	$s_{15} - s_{17}$
8	$s_{19} - s_{20}$
9	$s_{18} - s_{21}$
10	s_{16}

1) Teste 1: No primeiro teste, foi considerado que havia uma confiança inicial de 80% do robô estar no estado s_5 (os 20% restantes foram divididos uniformemente entre os demais estados) e que o mesmo tinha como destino o estado s_{16}. O resultado da seleção de ações com maior função valor $V(Bel, a)$ pode ser visto na Tabela IV, que apresenta os três maiores valores de $V(Bel, a)$ em cada época de decisão t. Como em $t = 1$, a ação a_{s_5,s_6} (ação que leva o robô de s_5 para s_6) apresenta maior função valor, a mesma é executada e a crença sobre os estados é atualizada usando o ES-MOM. O processo é repetido para as demais épocas de decisão, até que o critério de parada do Algoritmo 1 ($Bel(s_{16}) > 0.7$) seja atingido. As ações selecionadas na Tabela IV são ilustradas na Fig. 7 por meio de setas vermelhas. Elucidando assim os movimentos do robô durante o trajeto entre s_5 e s_{16}.

O ES-MOM convergiu para 100% de certeza sobre a localização do robô após a primeira iteração do Algoritmo 1, e manteve esta estimativa durante o trajeto. O caminho executado pelo robô foi (topologicamente) o ótimo. Um vídeo do experimento pode ser visto em https://youtu.be/67psg3uXHb8.

TABELA IV
AÇÕES SELECIONADAS ENTRE OS ESTADOS s_5 E s_{16}.

t	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$
1	$a_{s_5,s_6} \rightarrow 0.54$	$a_{s_5,s_1} \rightarrow 0.46$	-
2	$a_{s_6,s_7} \rightarrow 0.59$	$a_{s_6,s_5} \rightarrow 0.41$	-
3	$a_{s_7,s_8} \rightarrow 0.39$	$a_{s_7,s_3} \rightarrow 0.33$	$a_{s_7,s_6} \rightarrow 0.28$
4	$a_{s_8,s_{11}} \rightarrow 0.29$	$a_{s_8,s_4} \rightarrow 0.26$	$a_{s_8,s_{13}} \rightarrow 0.25$
5	$a_{s_{11},s_{12}} \rightarrow 0.49$	$a_{s_{11},s_4} \rightarrow 0.28$	$a_{s_{11},s_8} \rightarrow 0.23$
6	$a_{s_{12},s_{17}} \rightarrow 0.38$	$a_{s_{12},s_{10}} \rightarrow 0.26$	$a_{s_{12},s_{11}} \rightarrow 0.21$
7	$a_{s_{17},s_{16}} \rightarrow 0.48$	$a_{s_{17},s_{12}} \rightarrow 0.26$	$a_{s_{17},s_{10}} \rightarrow 0.26$

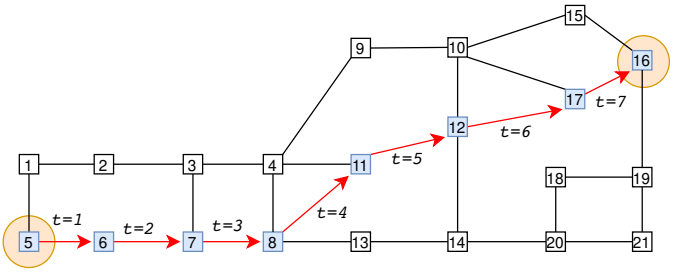


Fig. 7. Ilustração das ações selecionadas entre os estados s_5 e s_{16}.

2) Teste 2: Um segundo teste foi realizado, e os estados desejados foram o s_{12} e o s_1 (em sequência), para tal foi considerado que o robô partiu do estado s_{21} com uma confiança inicial de 30% (os 70% restantes foram divididos uniformemente entre os demais estados). Os resultados são exibidos na Tabela V e ilustrados na Fig. 8. Na primeira iteração do Algoritmo 1, há pequena margem de segurança para executar a ação $a_{s_{21},s_{20}}$ devido à baixa crença inicial. Na segunda iteração, o ES-MOM já apresenta boa confiança do robô ter ido para o estado s_{20} e a discrepância entre o maior e o segundo maior de $V(Bel, a)$ é aumentada. No restante do trajeto o ES-MOM mantém uma estimativa próxima a 100% sobre o verdadeiro estado ocupado pelo robô. O caminho executado do estado s_{21} ao estado s_1 foi (topologicamente) o ótimo, dada a restrição de passar previamente por s_{12}. Caso não houvesse a exigência de passar pelo estado s_{12}, o caminho ótimo poderia ser $[s_{21} - s_{20} - s_{14} - s_{13} - s_8 - s_4 - s_3 - s_2 - s_1]$, por exemplo. Um vídeo ilustrativo deste experimento pode ser visualizado em https://youtu.be/KGMhSLGTAiM.

TABELA V
AÇÕES SELECIONADAS ENTRE OS ESTADOS $s_{21} \rightarrow s_{12} \rightarrow s_1$.

t	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$
1	$a_{s_{21},s_{20}} \rightarrow 0.11$	$a_{s_{21},s_{19}} \rightarrow 0.09$	$a_{s_{10},s_{12}} \rightarrow 0.04$
2	$a_{s_{20},s_{14}} \rightarrow 0.46$	$a_{s_{20},s_{18}} \rightarrow 0.29$	$a_{s_{20},s_{21}} \rightarrow 0.25$
3	$a_{s_{14},s_{12}} \rightarrow 0.56$	$a_{s_{14},s_{13}} \rightarrow 0.24$	$a_{s_{14},s_{20}} \rightarrow 0.20$
4	$a_{s_{12},s_{11}} \rightarrow 0.39$	$a_{s_{12},s_{10}} \rightarrow 0.25$	$a_{s_{12},s_{14}} \rightarrow 0.22$
5	$a_{s_{11},s_4} \rightarrow 0.44$	$a_{s_{11},s_8} \rightarrow 0.30$	$a_{s_{11},s_{12}} \rightarrow 0.26$
6	$a_{s_4,s_3} \rightarrow 0.38$	$a_{s_4,s_8} \rightarrow 0.22$	$a_{s_4,s_{11}} \rightarrow 0.22$
7	$a_{s_3,s_2} \rightarrow 0.52$	$a_{s_3,s_4} \rightarrow 0.24$	$a_{s_3,s_7} \rightarrow 0.24$
8	$a_{s_2,s_1} \rightarrow 0.76$	$a_{s_2,s_3} \rightarrow 0.24$	-

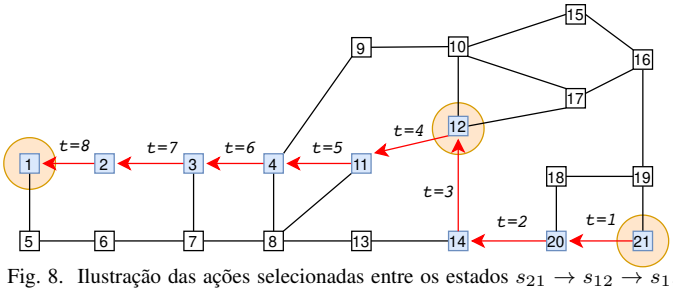


Fig. 8. Ilustração das ações selecionadas entre os estados $s_{21} \rightarrow s_{12} \rightarrow s_1$.

3) Teste 3: No terceiro teste, o objetivo era que o robô atingisse o estado s_{18}, porém não havia nenhuma informação sobre a sua localização inicial, o que implica em utilizar uma distribuição uniforme para representar a crença inicial Bel_0. Desta forma, o robô foi guiado inicialmente por um usuário usando um *joystick*, e conforme coletava informações das *landmarks* o ES-MOM atualizava as probabilidades $Bel(s')$. Quando o ES-MOM atingiu uma confiança maior que 70% de estar em um estado, o *joystick* foi desacoplado e o robô se dirigiu de forma autônoma para o estado s_{18} por meio do Algoritmo 1. O trajeto que o robô percorreu usando o *joystick* foi $[s_1 - s_2 - s_3 - s_4 - s_{11} - s_{12} - s_{17}]$, e é ilustrado na Fig. 9 (caminho com setas verdes). O resultado da estimação da localização neste percurso é visto na Tabela VI.

TABELA VI
PROBABILIDADE $Bel(s')$ DO ES-MOM PARA O TRAJETO $[s_1 - s_2 - s_3 - s_4 - s_{11} - s_{12} - s_{17}]$.

t	$a_{s,s'}$	Robô	$Bel(s')$	$Bel(s')$	$Bel(s')$
1	a_{s_1,s_2}	s_1	$s_1 = 0.50$	$s_5 = 0.50$	$s_{21} = 0.00$
2	a_{s_2,s_3}	s_2	$s_2 = 0.50$	$s_6 = 0.50$	$s_{21} = 0.00$
3	a_{s_3,s_4}	s_3	$s_3 = 0.50$	$s_7 = 0.50$	$s_{21} = 0.00$
4	$a_{s_4,s_{11}}$	s_4	$s_4 = 0.50$	$s_8 = 0.50$	$s_{21} = 0.00$
5	$a_{s_{11},s_{12}}$	s_{11}	$s_{11} = 0.50$	$s_{13} = 0.50$	$s_{21} = 0.00$
6	$a_{s_{12},s_{17}}$	s_{12}	$s_{12} = 0.53$	$s_{14} = 0.45$	$s_{10} = 0.02$
7	-	s_{17}	$s_{17} = 0.91$	$s_{15} = 0.06$	$s_{20} = 0.02$

O ES-MOM atingiu confiança maior que 70% na 7ª iteração (vide Tabela VI). A partir daí, o Algoritmo 1 entrou em funcionamento e o resultado da seleção de ações é visto na Tabela VII e ilustrado na Fig. 9 (caminho com setas vermelhas). Se o robô fosse desde o início de forma autônoma, certamente haveria alguma colisão (devido à ausência de informação inicial). Entretanto, usando uma aplicação semiautônoma, o alvo foi atingido com segurança. Um vídeo deste experimento pode ser visto em https://youtu.be/buqK3q86HAo.

TABELA VII
AÇÕES SELECIONADAS ENTRE OS ESTADOS s_{17} E s_{18}.

t	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$
7	$a_{s_{17},s_{16}} \rightarrow 0.44$	$a_{s_{17},s_{12}} \rightarrow 0.34$	$a_{s_{17},s_{10}} \rightarrow 0.22$
8	$a_{s_{16},s_{19}} \rightarrow 0.48$	$a_{s_{16},s_{17}} \rightarrow 0.26$	$a_{s_{16},s_{15}} \rightarrow 0.26$
9	$a_{s_{19},s_{18}} \rightarrow 0.61$	$a_{s_{19},s_{21}} \rightarrow 0.30$	$a_{s_{19},s_{16}} \rightarrow 0.09$

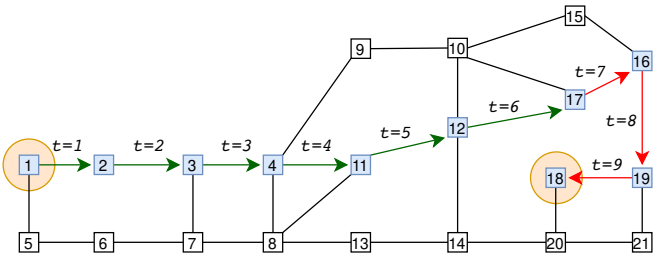


Fig. 9. Ilustração das ações selecionadas entre os estados $s_1 \rightarrow s_{17} \rightarrow s_{18}$.

V. CONCLUSÕES E TRABALHOS FUTUROS

Neste artigo, o PDMPO foi apresentado como uma alternativa para resolver o problema de planejamento das ações de um robô que se move por um espaço de convivência. O resultado da fase de planejamento e controle é a política, que determina as ações a serem tomadas pelo robô a fim de maximizar a recompensa esperada imediata. Por meio da otimização gananciosa e das simplificações utilizadas, o Algoritmo 1 resolve o PDMPO no espaço de estados e observações (em vez de resolver no espaço de crenças).

Na maioria das épocas de decisão há relativa discrepância entre o maior e o segundo maior valor de $V(Bel,a)$, possibilitando executar a ação $a_{s,s'}$ (com maior $V(Bel,a)$) com boa margem de segurança. Quando os valores da crença inicial são mais consistentes, a diferença entre o maior e o segundo maior valor de $V(Bel,a)$ é amplificada. Desta forma, o Algoritmo 1 se mostrou adequado para fazer o planejamento das ações do robô até o alvo, mesmo em um ambiente com pouca distinguibilidade, desde que se tenha algum conhecimento (mesmo que mínimo) sobre a configuração inicial.

Nos experimentos propostos, o planejador de movimento local do Apêndice A é o responsável por executar as políticas determinadas pelo PDMPO (ações em "em alto nível"). Entretanto, este planejador não considera a presença de obstáculos entre os estados. Portanto, almeja-se em trabalhos futuros incorporar ao planejador local: restrições de regiões proibidas no ambiente (obstáculos) e restrições nas ações de controle (velocidades máximas e mínimas que o robô poderá executar). A cada ciclo de execução do planejador local as informações de sensores (laser, por exemplo) seriam coletadas e montar-se-ia um problema de otimização, de modo a gerar uma trajetória que se direciona para o estado relacionado à política fornecida pelo PDMPO, desviando-se dos obstáculos presentes no ambiente. Assim, usando um método local deliberativo-reativo, obstáculos que não tinham sido previstos *a priori* poderiam ser adicionados ao problema de otimização, possibilitando uma navegação sem colisão com objetos dinâmicos (ou estáticos) distribuídos pelo ambiente, e sem movimentos bruscos.

APÊNDICE A
PLANEJAMENTO DE MOVIMENTO LOCAL

No instante que o Algoritmo 1 solicita uma ação $a_{s,s'}$ (que leva o robô do estado s para o s'), é desejado que ela seja executada em um tempo $\top$ (dado em função da distância entre s e s') e que o robô comece e termine o trajeto com velocidade zero (para poder observar as *landmarks*). Dado tais restrições, foi estabelecida uma trajetória para conectar os estados s e s':

$$\begin{bmatrix} x_\tau \\ y_\tau \end{bmatrix} = \begin{bmatrix} s_x \\ s_y \end{bmatrix} + \begin{bmatrix} s'_x - s_x \\ s'_y - s_y \end{bmatrix} \cdot \left(\frac{-2\tau^3}{\top^3} + \frac{3\tau^2}{\top^2} \right). \quad (19)$$

Em (19) é descrita a trajetória (desejada) a ser percorrida pelo robô para $\tau \in [0, \top]$. A referência de posição $(x_\tau, y_\tau)^T$ é seguida por um controlador *feedback linearization*, que fornecerá as velocidades $u_\tau = (v_\tau, \omega_\tau)^T$ que serão aplicadas à plataforma robótica (Fig. 3a) não holônomica:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \end{bmatrix} = \begin{bmatrix} \dot{x}_\tau \\ \dot{y}_\tau \end{bmatrix} + k \cdot \begin{bmatrix} x_\tau - x_R \\ y_\tau - y_R \end{bmatrix}, \quad (20)$$

$$\begin{bmatrix} v_\tau \\ \omega_\tau \end{bmatrix} = \begin{bmatrix} cos(\theta_R) & sen(\theta_R) \\ -sen(\theta_R)/d & cos(\theta_R)/d \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \end{bmatrix}. \quad (21)$$

O ganho do *feedback linearization* é representado por k e $(x_R, y_R, \theta_R)^T$ é a localização do robô estimada pelo EKF. A variável d simboliza um ponto próximo ao centro do robô, e tem a função evitar singularidades [5].

AGRADECIMENTOS

Os autores agradecem às agências CNPq, CAPES, e ao PPGEE da UFMG, pelo apoio financeiro.

REFERÊNCIAS

[1] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.
[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
[3] Y. A. Kapitanyuk, A. V. Proskurnikov, and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1372–1385, 2017.
[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
[5] N. Monteiro, C. Maia, and V. Gonçalves, "Controle de um robô móvel em um galpão de estoque utilizando processo de decisão de markov parcialmente observável," in *Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI)*, 2019.
[6] J. Pellegrini and J. Wainer, "Processos de decisão de markov: um tutorial," *Revista de Informática Teórica e Aplicada*, vol. 14, no. 2, pp. 133–179, 2007.
[7] J. Pineau, G. Gordon, and S. Thrun, "Tractable planning under uncertainty: exploiting structure," Ph.D. dissertation, Carnegie Mellon University, the Robotics Institute, 2004.
[8] S. Koenig and R. Simmons, "Xavier: A robot navigation architecture based on partially observable markov decision process models," *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, no. partially, pp. 91–122, 1998.
[9] E. Páll, L. Tamás, and L. Buşoniu, "Analysis and a home assistance application of online aems2 planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5013–5019.
[10] C. Wang, J. Cheng, J. Wang, X. Li, and M. Q.-H. Meng, "Efficient object search with belief road map using mobile robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3081–3088, 2018.
[11] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, W. L. LS, and S. Tellex, "Multi-object search using object-oriented pomdps," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7194–7200.
[12] W. V. Arantes and E. A. L. Junior, "A strategy for the use of indoor and outdoor augmented reality location systems applied to smartphones," *IEEE Latin America Transactions*, vol. 16, no. 5, pp. 1460–1467, 2018.
[13] D. Aberdeen, "A survey of approximate methods for solving partially observable markov decision process," *Report National ICT*, 2003.
[14] D. Braziunas, "Pomdp solution methods," *University of Toronto*, 2003.
[15] J. Pineau and G. J. Gordon, "Pomdp planning for robust robot control," in *Robotics Research*. Springer, 2007, pp. 69–82.
[16] I. Kostavelis, D. Giakoumis, S. Malassiotis, and D. Tzovaras, "A pomdp design framework for decision making in assistive robots," in *International Conference on Human-Computer Interaction*. Springer, 2017, pp. 467–479.
[17] G. E. de Andrade, L. A. P. L. Junior, A. Calsavara, G. A. Michelon, and V. Brussamolin, "A greedy routing strategy based on euclidean geometry for vehicular delay tolerant network," *IEEE Latin America Transactions*, vol. 16, no. 7, pp. 2000–2006, 2018.
[18] V. Krishnamurthy, *Partially observed Markov decision processes*. Cambridge University Press, 2016.
[19] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.
[20] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.
[21] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems," in *AAAI/IAAI*, 1999, pp. 541–548.
[22] Y. Bengio and P. Frasconi, "Input-output hmms for sequence processing," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1231–1249, 1996.
[23] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Readings in speech recognition*. Elsevier, 1990, pp. 267–296.
[24] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *Journal of artificial intelligence research*, vol. 13, pp. 33–94, 2000.
[25] A. Foka and P. Trahanias, "Real-time hierarchical pomdps for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, 2007.
[26] G. V. Lima, R. M. J. A. de Souza, A. S. de Morais, L. C. Oliveira-Lopes, and G. M. V. Ladeira, "Stabilization and path tracking of a mini quadrotor helicopter: Experimental results," *IEEE Latin America Transactions*, vol. 17, no. 03, pp. 485–492, 2019.
[27] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

Neemias Silva Monteiro Possui graduação em Engenharia de Controle e Automação pela Universidade Federal de Itajubá (2015) e mestrado em Engenharia Elétrica pela Universidade Federal de Minas Gerais (2020). Tem experiência na área de localização e planejamento de movimento de robôs; e com modelos markovianos.

Vinicius Mariano Gonçalves Recebeu o título de bacharel em Engenharia de Controle e Automação pela Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brasil, em 2011, e de doutor em Engenharia elétrica pela mesma universidade em 2014. Ele está associado à Escola de Engenharia, UFMG, desde 2016, onde é Professor de Engenharia Elétrica. Seus interesses atuais de pesquisa incluem controle de sistemas dinâmicos orientados por eventos discretos, robótica, e otimização.

Carlos Andrey Maia Recebeu os títulos de bacharel e mestre em Engenharia Elétrica pela Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brasil em 1994 e 1996 respectivamente. Os títulos de doutor pela Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, e pela Université d'Angers, Angers, França, foram obtidos em 2003, graças a um projeto de cotutela. Atualmente é Professor Titular do Departamento de Engenharia Elétrica da Escola de Engenharia da UFMG, onde ingressou em 1997. Seus interesses atuais de pesquisa incluem modelagem, análise de desempenho e controle de sistemas, com destaque para aqueles cuja dinâmica é orientada por eventos discretos.