

Laboratório de Sistemas de Computação e Robótica
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627, CEP 31270-901,
Belo Horizonte, MG, Brasil



DISSERTAÇÃO DE MESTRADO Nº 1172

Localização e Planejamento de Movimento de
Robôs Móveis em Ambientes Internos Utilizando
Processos de Decisão de Markov

Neemias Silva Monteiro

Belo Horizonte
18 de fevereiro de 2020

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LOCALIZAÇÃO E PLANEJAMENTO DE MOVIMENTO
DE ROBÔS MÓVEIS EM AMBIENTES INTERNOS
UTILIZANDO PROCESSOS DE DECISÃO DE MARKOV

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Neemias Silva Monteiro

Orientador: Prof. Dr. Carlos Andrey Maia

Coorientador: Prof. Dr. Vinicius Mariano Gonçalves

Belo Horizonte

18 de fevereiro de 2020

M775I

Monteiro, Neemias Silva.

Localização e planejamento de movimento de robôs móveis em ambientes internos utilizando processos de decisão de Markov [recurso eletrônico] / Neemias Silva Monteiro. – 2020.

1 recurso online (106 f. : il., color.) : pdf.

Orientador: Carlos Andrey Maia.

Coorientador: Vinicius Mariano Gonçalves.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 87-98.

Bibliografia: f. 99-106.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Incerteza - Teses. 3. Robótica - Teses. 4. Modelo oculto de Markov - Teses. I. Maia, Carlos Andrey. II. Gonçalves, Vinicius Mariano. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)


**Localização e Planejamento de Movimento de Robôs Móveis
em Ambientes Internos Utilizando Processos de Decisão de
Markov**

Neemias Silva Monteiro

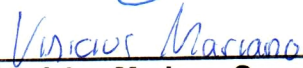
Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 18 de fevereiro de 2020.

Por:



Prof. Dr. Carlos Andrey Maia
DEE (UFMG) - Orientador



Prof. Dr. Vinicius Mariano Gonçalves
DEE (UFMG) - Coorientador



Prof. Dr. Guilherme Augusto Silva Pereira
DEE (UFMG)



Prof. Dr. Gustavo Medeiros Freitas
DEE (UFMG)



Prof. Dr. Armando Alves Neto
DELT (UFMG)

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me sustentado e guiado até este momento.

Agradeço aos meus orientadores Prof. Carlos Maia e Prof. Vinicius Gonçalves pelos ensinamentos e pela paciência durante o mestrado. Agradeço aos membros da banca examinadora, Prof. Armando Neto, Prof. Guilherme Pereira e Prof. Gustavo Freitas, pelas valiosas sugestões que contribuíram para o aprimoramento desta dissertação.

Gostaria de agradecer aos colegas do laboratório CORO pelo companheirismo e amizade, em especial ao Adriano e ao Victor pela ajuda nos experimentos.

Também sou grato aos professores do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) pelo aprendizado durante as disciplinas.

Finalmente, agradeço ao CNPq e ao PPGEE pelo apoio financeiro.

*“Porque a graça de Deus se há manifestado, trazendo salvação a todos os homens”
(Tt 2.11)*

RESUMO

NEEMIAS S. MONTEIRO. **Localização e Planejamento de Movimento de Robôs Móveis em Ambientes Internos Utilizando Processos de Decisão de Markov**. 2020. 106 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

Os planejadores de movimento determinísticos apresentam bons resultados em ambientes simulados, nos quais os sensores e atuadores são perfeitos. No entanto, estas suposições são restritivas e conseqüentemente o planejamento de movimento terá desempenho comprometido se aplicado em sistemas robóticos reais (ou em um simulador mais realístico), pois os mesmos são inerentemente carregados de incertezas. Na maioria dos sistemas robóticos reais, os estados não podem ser diretamente observados, e as medições recebidas pelo robô são projeções ruidosas do verdadeiro estado. As ações executadas por um robô também têm incertezas, dado que os atuadores do robô cometem erros ao seguir os comandos de controle desejados. Deste modo, o robô deve fazer uso de uma nova classe de planejadores que levam em conta as incertezas do sistema quando tiver que tomar uma decisão.

No presente trabalho, o Processo de Decisão de Markov Parcialmente Observável (PDMPO, ou POMDP em inglês), é apresentado como alternativa para solucionar problemas imersos em incertezas, selecionando ações ótimas com objetivo de realizar uma dada tarefa. O PDMPO é um método probabilístico que considera: que os estados do robô não podem ser medidos diretamente, mas são inferidos por meio de observações indiretas; que as decisões tomadas têm resultados incertos; e que o resultado de uma ação em um estado depende apenas da ação e do estado atual do processo (propriedade Markoviana). No PDMPO cada ação tem como resultado uma observação, probabilisticamente relacionada aos estados do sistema. Em vez de um estado atual do sistema, no PDMPO há uma distribuição de probabilidade sobre os estados, denominada crença. Para estimativa da crença, este trabalho utiliza a estrutura probabilística do Modelo Oculto de Markov (MOM, ou HMM em inglês).

O ferramental acima foi aplicado em um sistema simulado de localização e controle das ações de um robô que se locomove por um galpão usado para estocagem de produtos, e também para navegação de um robô real em um espaço de convivência. As simulações e experimentos mostram a robustez e eficiência dos métodos utilizados.

Palavras-chave: Incertezas, Localização, Modelo Oculto de Markov, Planejamento de Movimento, Processo de Decisão de Markov Parcialmente Observável, Robótica Probabilística.

ABSTRACT

NEEMIAS S. MONTEIRO. **Localização e Planejamento de Movimento de Robôs Móveis em Ambientes Internos Utilizando Processos de Decisão de Markov**. 2020. 106 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

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 measurements received by the robot are noisy projections of the true state. The actions performed by a robot have uncertainties, given that the robot's actuators make mistakes when following the desired control commands. 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 (POMDP) is presented as an alternative to solve problems immersed in uncertainties, selecting optimal actions aiming to perform a given task. POMDP is a probabilistic method that considers: that robot states cannot be measured directly, but are inferred through indirect observations; that the decisions taken have uncertain results; and that the result of an action in a state depends only on the action and the current state of the process (Markovian property). In POMDP, each action results in an observation, probabilistically related to states of the system. Instead of a current system state, in POMDP there is a probability distribution over states, called belief. To estimate the belief, this work used the probabilistic structure of the Hidden Markov Model (HMM).

The above methodology was applied to a simulated system for localization and controlling the actions of a robot that moves around a warehouse used for products storage, as well as for navigating a real robot in a living space. Simulations and experiments show the robustness and efficiency of the methods used.

Keywords: Uncertainty, Localization, Hidden Markov Model, Motion Planning, Partially Observable Markov Decision Process, Probabilistic Robotics.

LISTA DE ILUSTRAÇÕES

Figura 1.1 – Ambientes internos grandes, simétricos e ambíguos.	19
Figura 2.1 – Planejamento do caminho para conectar a configuração q_{init} à q_{goal}	23
Figura 2.2 – Modelo gráfico para localização de robôs móveis. O objetivo da localização é inferir a verdadeira pose do robô x , a partir das variáveis conhecidas: mapa m , medições z e controles u	24
Figura 2.3 – Exemplos de aplicações com robô móvel utilizando <i>landmarks</i> para se localizar no ambiente.	25
Figura 2.4 – Comparação entre o desempenho das câmeras produzidas com a tecnologia CCD e CMOS, observa-se que nos próximos anos a CMOS será dominante no mercado	27
Figura 2.5 – Modelo de cores RGB e HSV.	27
Figura 2.6 – Exemplos de câmeras que extraem informação 3D de uma cena.	28
Figura 2.7 – A ilustração da esquerda mostra a imagem de um ambiente capturada com o módulo RGB. Já a ilustração da direita, mostra a imagem de profundidade, na qual os pontos mais próximos são mostrados com cores azuis e os mais distantes com cores vermelhas. Os objetos de plano de fundo, que ultrapassam o alcance do sensor, são representados em azul escuro	28
Figura 2.8 – Mapeamento com poses conhecidas.	29
Figura 2.9 – Exemplos de mapas utilizados para navegação de robôs.	30
Figura 2.10–Modelo gráfico do <i>online SLAM</i> e <i>full SLAM</i>	31
Figura 2.11–Exemplo de <i>loop closure</i> . O trajeto azul indica um caminho típico sem utilizar o <i>loop closure</i> . O caminho vermelho mostra o resultado da utilização do <i>loop closure</i> após detectar um local já visitado	31
Figura 3.1 – Representação gráfica de uma cadeia de Markov de três estados.	34
Figura 3.2 – Modelo gráfico simplificado para o MOM	35
Figura 3.3 – Modelo gráfico detalhado para o MOM	36
Figura 3.4 – Procedimento <i>forward</i> : estado s_t em função do estado s_{t-1}	39
Figura 3.5 – Procedimento <i>backward</i> : estado s_t em função do estado s_{t+1}	41
Figura 3.6 – Modelo gráfico simplificado para o ES-MOM	44
Figura 3.7 – Esboço da vista superior de um robô com acionamento diferencial	49
Figura 3.8 – Plataforma robótica utilizada nos experimentos reais.	50

Figura 3.9 – Visualização do funcionamento do ORB-SLAM 2 no Rviz. Na imagem à esquerda, temos as <i>features</i> identificadas no ambiente pela câmera RGB-D (destacadas pelos quadrados verdes). Na imagem à direita, temos o mapa de <i>features</i> e o sistema de referência da câmera RGB-D.	51
Figura 3.10–Marcador fiducial Aruco	52
Figura 3.11–Transformações necessárias para encontrar a pose do robô quando um Aruco é identificado.	53
Figura 3.12–Esquemático do funcionamento do Filtro de Kalman Estendido.	54
Figura 3.13–Localização da plataforma robótica em corredores usando câmera RGB-D + ORB-SLAM 2.	54
Figura 3.14–Localização da plataforma robótica em corredores usando a câmera <i>Tracking</i>	55
Figura 3.15–Mapa do ambiente topológico utilizado para simulação do MOM.	56
Figura 3.16–Modelo gráfico do ES-MOM incorporando o estado virtual \bar{s} , utilizado para calcular a probabilidade de transição de estados $p(q_t = s_j q_{t-1} = s_i, \mathbb{U}_{t-1})$	59
Figura 3.17–Mapa topológico (em escala real) utilizado nos experimentos do ES-MOM.	62
Figura 4.1 – Modelo gráfico simplificado para o PDM	67
Figura 4.2 – Representação gráfica para o PDMPO	71
Figura 4.3 – Esquemático de funcionamento do PDMPO utilizado neste trabalho.	74
Figura 4.4 – Sistema de estocagem proposto, desenvolvido no <i>software</i> V-REP.	76
Figura 4.5 – Integração do sistema desenvolvido utilizando o <i>framework</i> ROS.	77
Figura 4.6 – Mapa topológico do sistema proposto.	77
Figura 4.7 – Perspectiva do espaço de convivência utilizado nos experimentos do PDMPO. Os retângulos verdes representam sofás, os laranjas <i>poofs</i> , e o cinza uma escada. As conexões entre os 21 estados do ambiente são indicadas pelas linhas azuis.	81
Figura 4.8 – Espaço de convivência utilizado nos experimentos do PDMPO.	82
Figura A.1–Gráfico do polinômio auxiliar $g(t)$ e sua derivada primeira $\dot{g}(t)$ para $\top = 10$	88
Figura B.1 –Evolução dos estados $(x, y, \theta)^T$ pertencentes às trajetórias ótimas.	93
Figura B.2–Saídas de controle que devem ser aplicadas para se atingir as trajetórias ótimas.	93
Figura B.3–Custo ótimo para 50 iterações da PDD.	94
Figura B.4–Influência do parâmetro κ na penalização de um obstáculo, utilizando $x_0 = (1, 1)^T$ e $x_T = (9, 1)^T$	96
Figura B.5–Sinal de controle \hat{u} para o teste apresentado na Figura B.4.	97
Figura B.6–Evolução dos estados $(x, y)^T$ pertencentes à trajetória ótima em ambientes com obstáculos.	97
Figura B.7–Saídas de controle que devem ser aplicadas para se atingir as trajetórias ótimas.	98

LISTA DE ALGORITMOS

Algoritmo 1 – PROCEDIMENTO FORWARD	40
Algoritmo 2 – PROCEDIMENTO BACKWARD	42
Algoritmo 3 – VITERBI	44
Algoritmo 4 – FILTRO DE BAYES	46
Algoritmo 5 – FILTRO DE KALMAN	47
Algoritmo 6 – FILTRO DE KALMAN ESTENDIDO	48
Algoritmo 7 – ITERAÇÃO DE VALORES PARA PDM	70
Algoritmo 8 – RESOLUÇÃO DO PDMPO USANDO OTIMIZAÇÃO “GANANCIOSA”	75

LISTA DE TABELAS

Tabela 1.1 – Família dos modelos de Markov.	18
Tabela 3.1 – Símbolos de observações referentes a cada estado (simulação).	56
Tabela 3.2 – Probabilidade α para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).	57
Tabela 3.3 – Probabilidade β para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).	57
Tabela 3.4 – Probabilidade γ para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).	57
Tabela 3.5 – Probabilidade α para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).	58
Tabela 3.6 – Probabilidade β para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).	58
Tabela 3.7 – Probabilidade γ para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).	58
Tabela 3.8 – Probabilidade α do ES-MOM para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com σ).	60
Tabela 3.9 – Probabilidade α do ES-MOM para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com 0.1σ).	60
Tabela 3.10–Probabilidade α do ES-MOM para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com 0.01σ).	60
Tabela 3.11–Probabilidade α do ES-MOM para o trajeto $[s_{30} - s_{32} - s_{17} - s_{27} - s_{14} - s_{22} - s_{23} - s_{39}]$ (com σ).	61
Tabela 3.12–Símbolos de observações (id's) referentes a cada estado (experimento).	61
Tabela 3.13–Probabilidade α do ES-MOM para o trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$	63
Tabela 3.14–Probabilidade α do MOM para o trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$	63
Tabela 3.15–Probabilidade α do ES-MOM para o trajeto $[s_{15} - s_{14} - s_{13} - s_{12} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_3 - s_2 - s_1]$ com oclusões das <i>landmarks</i> nos estados s_4 e s_{37}	64
Tabela 4.1 – Observações referentes a cada estado.	78
Tabela 4.2 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (usando a Equação (4.28)).	79

Tabela 4.3 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (usando a Equação (4.29)).	80
Tabela 4.4 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (considerando uma crença inicial de 30% do robô estar no estado s_3).	80
Tabela 4.5 – Ações selecionadas entre os estados $s_{33} \rightarrow s_{21} \rightarrow s_{23} \rightarrow s_8$	81
Tabela 4.6 – Símbolos de observações (id's) referentes a cada estado (experimento do PDMPO).	82
Tabela 4.7 – Ações selecionadas entre os estados s_5 e s_{16} , para cada época de decisão t	82
Tabela 4.8 – Ações selecionadas entre os estados $s_{21} \rightarrow s_{12} \rightarrow s_1$, para cada época de decisão t	83
Tabela 4.9 – Probabilidade α do ES-MOM para o trajeto $[s_1 - s_2 - s_3 - s_4 - s_{11} - s_{12} - s_{17}]$	83
Tabela 4.10–Ações selecionadas entre os estados s_{17} e s_{18} , para cada época de decisão t	84
Tabela B.1 – Parâmetros dos testes com a PDD (sem restrições).	93

LISTA DE ABREVIATURAS E SIGLAS

CCD	<i>Charge Coupled Device</i>
CMOS	<i>Complementary Metal Oxide Semiconductor</i>
ES-MOM	Entrada Saída - Modelo Oculto de Markov
FK	Filtro de Kalman
FKE	Filtro de Kalman Estendido
GPS	<i>Global Positioning System</i>
HMM	<i>Hidden Markov Model</i>
HSV	<i>Hue Saturation Value</i>
IMU	<i>Inertial Measurement Unit</i>
MOM	Modelo Oculto de Markov
PD	Programação Dinâmica
PDD	Programação Dinâmica Diferencial
PDF	<i>Probability Density Function</i>
PDM	Processo de Decisão de Markov
PDMPO	Processo de Decisão de Markov Parcialmente Observável
POMDP	<i>Partially Observable Markov Decision Processes</i>
PRM	<i>Probabilistic Roadmap</i>
RGB	<i>Red Green Blue</i>
RGB-D	<i>Red Green Blue - Depth</i>
ROS	<i>Robot Operating System</i>
RRT	<i>Rapidly-Exploring Random Tree</i>
RTDP	<i>Real Time Dynamic Programming</i>
RUR	<i>Rossum's Universal Robots</i>
SLAM	<i>Simultaneous Localization and Mapping</i>
UFMG	Universidade Federal de Minas Gerais
V-REP	<i>Virtual - Robot Experimentation Platform</i>

SUMÁRIO

Lista de ilustrações	viii	
Lista de algoritmos	x	
Lista de tabelas	xi	
Lista de abreviaturas e siglas	xiii	
Sumário	xiv	
1	INTRODUÇÃO	17
1.1	Definição do problema	19
1.2	Motivação	19
1.3	Objetivos	20
1.4	Publicações	20
1.5	Estrutura da dissertação	21
2	REVISÃO BIBLIOGRÁFICA	22
2.1	Problemas fundamentais da robótica	22
2.2	Planejamento de caminho	23
2.3	Localização	24
2.4	Percepção	26
2.5	Mapeamento	29
2.6	SLAM	30
3	LOCALIZAÇÃO DE ROBÔS MÓVEIS TERRESTRES EM AMBI- ENTES INTERNOS	32
3.1	Localização de robôs em ambientes topológicos	33
3.1.1	<i>Cadeia de Markov</i>	33
3.1.2	<i>Modelo Oculto de Markov (MOM)</i>	34
3.1.3	<i>MOM: trabalhos relacionados</i>	36
3.1.4	<i>Os três problemas do MOM</i>	37
3.1.4.1	<i>Solução do Problema 3.1</i>	38
3.1.4.2	<i>Solução do Problema 3.2</i>	43
3.1.5	<i>ES-MOM</i>	44

3.1.6	<i>Resumo da Seção 3.1</i>	45
3.2	Localização de robôs em espaços contínuos	45
3.2.1	<i>Filtro de Kalman</i>	46
3.2.2	<i>Filtro de Kalman Estendido</i>	48
3.2.2.1	<i>Modelo cinemático do robô diferencial</i>	49
3.3	Simulações e Experimentos	50
3.3.1	<i>Plataforma robótica</i>	50
3.3.2	<i>Experimento com o Filtro de Kalman Estendido</i>	50
3.3.3	<i>Modelo Oculto de Markov</i>	55
3.3.3.1	<i>Simulação com um robô pontual</i>	55
3.3.3.1.1	MOM	57
3.3.3.1.2	ES-MOM	59
3.3.3.2	<i>Experimento com robô real (ES-MOM)</i>	61
3.4	Conclusões do capítulo	64
4	INTEGRAÇÃO ENTRE LOCALIZAÇÃO E PLANEJAMENTO DE MOVIMENTO DE ROBÔS EM AMBIENTES INTERNOS	65
4.1	Processo de Decisão de Markov (PDM)	66
4.1.1	<i>Conceitos</i>	68
4.1.2	<i>Algoritmo de iteração de valores</i>	70
4.2	Processo de Decisão de Markov Parcialmente Observável (PDMPO)	71
4.3	Simulação de controle de um robô em um galpão de estoque de produtos usando o PDMPO	76
4.3.1	<i>Mapa Topológico</i>	77
4.3.2	<i>Resultados e discussões</i>	78
4.3.2.1	<i>Teste 1</i>	79
4.3.2.2	<i>Teste 2</i>	80
4.4	Experimento em um espaço de convivência usando o PDMPO	81
4.4.1	<i>Teste 1</i>	82
4.4.2	<i>Teste 2</i>	83
4.4.3	<i>Teste 3</i>	83
4.5	Conclusões do capítulo	84
5	CONCLUSÕES E TRABALHOS FUTUROS	85
APÊNDICE A	PLANEJADOR DE MOVIMENTO LOCAL	87
APÊNDICE B	PLANEJADOR DE MOVIMENTO LOCAL COM RESTRIÇÕES DE CONTROLE E OBSTÁCULOS	89
B.1	Programação Dinâmica Diferencial	90

B.1.1	<i>Procedimento backward</i>	91
B.1.2	<i>Propagação de movimento</i>	92
B.2	Geração de trajetórias sem restrições	92
B.3	Geração de trajetórias com restrições de controle e obstáculos . . .	94
	Referências	99

INTRODUÇÃO

Desde os primórdios, o homem sempre tem buscado maneiras de aprimorar seus processos produtivos, tornando-os mais eficientes. Uma importante solução encontrada para tal foi a criação de sistemas robóticos. O termo robô tem origem na palavra tcheca ‘*robota*’, que significa ‘trabalhador forçado’. O termo foi apresentado pela primeira vez em 1921 na cidade de Praga, pelo dramaturgo Karel Capek’s na peça teatral RUR (*Rossum’s Universal Robots*). A peça conta a história do inventor Rossum, que desenvolve uma criatura com forma humana capaz de substituir o ser humano em qualquer tipo de trabalho [Murphy, 2000b]. A partir desta peça, o termo passou a ser usado em outros espetáculos e filmes de ficção científica. Entretanto, o avanço da tecnologia permitiu que os sistemas robóticos móveis deixassem de ser uma utopia da ficção, e passassem a serem desenvolvidos em grandes escalas e utilizados em várias aplicações.

De acordo com Dudek and Jenkin [2010], os robôs móveis podem ser classificados com relação ao meio de locomoção em: terrestres, aquáticos, aéreos e espaciais. Um robô é tipicamente constituído por uma combinação de componentes físicos (fontes de energia, mecanismos de locomoção e sensores) e computacionais (que controlam o robô). A utilização de sistemas robóticos móveis vai desde aplicações domésticas às industriais. Em geral, os robôs são requisitados em situações que o ambiente de trabalho é: (i) inóspito, ou que possa trazer riscos à saúde de um operador; (ii) remoto, no qual o envio de seres humanos é caro e demanda muito tempo; e (iii) repetitivo, de modo a causar fadiga ao operador.

A robótica clássica assume que os sensores de um robô podem mensurar completamente o estado do ambiente, e que os efeitos das ações de controle são determinísticos. Entretanto, estas suposições são utópicas, pois na maioria das aplicações práticas os estados não podem ser diretamente observados, e as medições recebidas pelo robô são projeções ruidosas do estado. As ações executadas por um robô também têm incertezas, dado que os resultados das ações são não determinísticos. Deste modo, o robô deve levar em conta as incertezas do ambiente e das suas

ações quando tiver que tomar uma decisão [Thrun et al., 2005]. Em muitas situações, o ambiente por onde os robôs se locomovem é imprevisível e dinâmico, implicando em grandes incertezas durante a navegação. A família dos modelos Markov fornece um arcabouço para resolver estes problemas robóticos estocásticos. Os modelos de Markov podem ser utilizados em diferentes situações, dependendo se o estado é totalmente observável e se é alterado com a tomada de decisão. Abaixo são mostradas variações do modelo de Markov discutidas neste trabalho.

- **Cadeia de Markov:** Modela um sistema totalmente observável, e são estabelecidas probabilidades de transição entre os estados. Não é possível tomar decisões.
- **Modelo Oculto de Markov (MOM):** É um modelo de Markov no qual os estados não são diretamente observáveis, ou seja, ocultos. Na cadeia de Markov considera-se que cada estado corresponde a uma observação, já o MOM representa o cenário em que a observação é uma distribuição de probabilidade sobre os estados. A tomada de decisão também não é explicitada. Utilizando os procedimentos “*forward*” e “*backward*” calcula-se, a cada iteração, a localização mais provável do sistema.
- **Processo de Decisão de Markov (PDM):** Assim como na cadeia de Markov, os estados são totalmente observáveis. Porém, há possibilidade de um agente intervir no sistema tomando decisões (executando ações), e cada decisão possui um resultado incerto. Para cada ação é atribuída uma recompensa, que depende do estado que o sistema se encontra. Para resolver um PDM deve-se encontrar a política (mapeamento de estados em ações) ótima, que determina qual decisão tomar para maximizar a recompensa esperada da sequência de decisões. Em suma, no PDM o modelo de percepção é determinístico e o modelo de transição de estados é estocástico.
- **Processo de Decisão de Markov Parcialmente Observável (PDMPO):** A diferença do PDMPO para o PDM, é que no PDMPO o estado do sistema não é conhecido quando decisões são tomadas. As informações disponíveis para o tomador de decisões são: as observações resultantes de cada decisão e a sequência de ações já tomadas.

Uma síntese dos modelos de Markov é mostrada na Tabela 1.1.

Tabela 1.1 – Família dos modelos de Markov.

Modelos de Markov		Incerteza no sensor?	
		Não	Sim
Problema de controle?	Não	Cadeia de Markov	Modelo Oculto de Markov - MOM
	Sim	Processo de Decisão de Markov PDM	Processo de Decisão de Markov Parcialmente Observável - PDMPO

1.1 Definição do problema

O problema que esta dissertação trata é estruturado da seguinte maneira:

Problema: *Estimar a localização de um robô móvel terrestre em um ambiente topológico interno conhecido usando marcos visuais distribuídos pelo ambiente. E a partir da estimativa de localização, determinar políticas de controle ótimas a serem executadas pelo robô com o propósito de alcançar um local predeterminado do ambiente.*

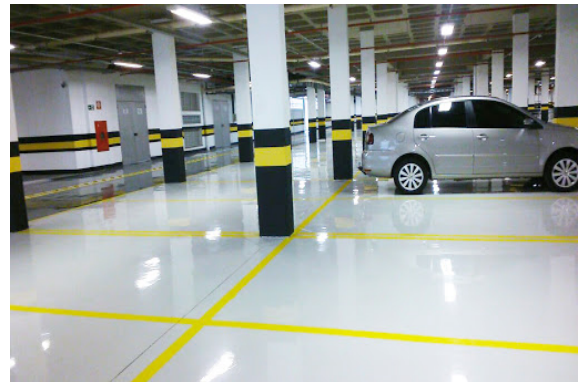
Para modelar as incertezas da localização do robô móvel faremos uso do MOM. Em vez de estabelecer uma única hipótese (único estado) de localização para o robô, o MOM fornecerá uma distribuição de probabilidade sobre todos os estados possíveis. Baseado na estimativa de localização do MOM, utilizaremos a estrutura do PDMPO para estabelecer políticas de controle a fim de que o robô atinja um estado desejado.

1.2 Motivação

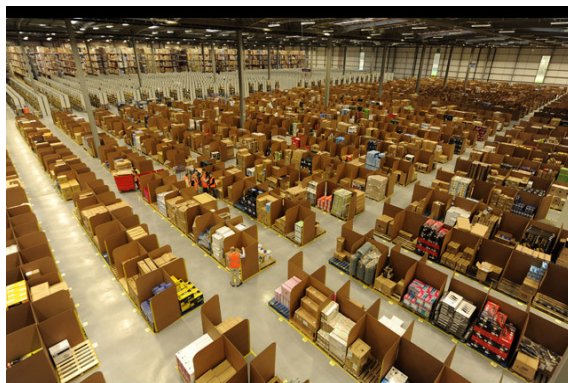
As estratégias tratadas nesta dissertação podem ser aplicadas em problemas de localização e tomada de decisão de robôs móveis em ambientes internos grandes, simétricos e ambíguos, como os mostrados na Figura 1.1. Nestes locais, a tomada de decisão se torna mais difícil devido à indistinguibilidade entre várias partes do ambiente, e a falta de acesso às medições de GPS.



(a) Corredor de hospital.



(b) Estacionamento de *shopping*.



(c) Armazém de estocagem de produtos.



(d) Biblioteca.

Figura 1.1 – Ambientes internos grandes, simétricos e ambíguos.

Apesar da indistinguibilidade presente nestes ambientes, o MOM consegue contorná-las e fornecer uma boa estimativa de localização para que a tomada de decisão (com o PDMPO) seja executada de forma coerente e com uma margem de segurança aceitável.

1.3 Objetivos

O principal objetivo deste trabalho é integrar a localização e planejamento de movimento por meio da estrutura do PDMPO, a fim de selecionar ações de controle adequadas para navegação de robôs em ambientes topológicos. Os objetivos específicos são listados abaixo:

1. Implementar um sistema de localização robusto para ambientes topológicos internos usando o MOM. Nas aplicações propostas consideramos que os sistemas são ambíguos e simétricos, e que não há conhecimento prévio sobre a localização inicial.
2. Implementar o PDMPO com baixo custo computacional através da otimização “gananciosa”, uma vez que os métodos (de solução) tradicionais têm alta complexidade computacional e aplicação limitada a problemas pequenos.
3. Implementar um gerador de trajetórias ótimas, sob restrições de controle e obstáculos, para que o robô possa percorrer o trajeto entre os estados do mapa topológico. O método utilizado para gerar as trajetórias é a Programação Dinâmica Diferencial (PDD).
4. Validar as estratégias propostas por meio de simulações e experimentos com um robô real.

1.4 Publicações

A principal contribuição deste trabalho é utilizar o MOM na fase de estimação de estados do PDMPO. Desta forma, a tomada de decisão é realizada a partir de uma estimativa de localização robusta. Esta dissertação contribuiu com três artigos publicados em congressos nacionais:

- [[Monteiro et al., 2019a](#)] Monteiro, N. S., Maia, C. A., Gonçalves, V. M. Controle de um robô móvel em um galpão de estoque utilizando Processo de Decisão de Markov Parcialmente Observável (PDMPO). In: Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI), 2019, Ouro Preto.
 - Neste artigo, foi simulado um sistema de controle das ações de um robô (usando o PDMPO) que se locomove por um galpão utilizado para estocagem de produtos.
- [[Monteiro et al., 2019b](#)] Monteiro, N. S., Maia, C. A., Gonçalves, V. M. Estimação da localização de um robô não holonômico utilizando Modelo Oculto de Markov (MOM) e o Filtro de Kalman Estendido (FKE). In: Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI), 2019, Ouro Preto.

- Neste artigo, utiliza-se o MOM e o FKE para estimar a posição de um robô móvel não holonômico durante o rastreamento de trajetórias parametrizadas no tempo.
- [Monteiro et al., 2019c] Monteiro, N. S., Maia, C. A., Gonçalves, V. M. Geração de trajetórias ótimas para o robô não holonômico usando Programação Dinâmica Diferencial (PDD). In: Anais da 14^a Conferência Brasileira de Dinâmica Controle e Aplicações (DINCON), 2019, São Carlos.
 - Neste artigo é apresentada a PDD, método de geração de trajetórias ótimas para um robô não holonômico, dado um funcional de custo quadrático.

1.5 Estrutura da dissertação

O Capítulo 1 apresentou as motivações e as principais contribuições da dissertação. No Capítulo 2, fazemos uma revisão da literatura descrevendo alguns problemas recorrentes da robótica. O Capítulo 3 discute métodos de localização de robôs em ambientes internos definidos topologicamente, fazendo uso de uma variação do MOM, o ES-MOM. O Capítulo 4 discorre sobre a estrutura do PDMPO e a formulação encontrada para realização da otimização “gananciosa”. Ao final dos Capítulos 3 e 4, são mostrados experimentos reais com uma plataforma robótica e algumas simulações nos *softwares* Matlab e V-REP. No Capítulo 5, serão apresentadas as conclusões e as propostas para trabalhos futuros.

O Apêndice A mostra o método utilizado neste trabalho para executar as políticas do PDMPO: geração de trajetórias usando funções polinomiais sem considerar restrições de controle e obstáculos. Já o Apêndice B, exibe uma alternativa para executar as políticas do PDMPO, sob restrições de controle e obstáculos, usando a PDD.

REVISÃO BIBLIOGRÁFICA

Neste capítulo, é apresentada uma revisão dos principais problemas relacionados à robótica móvel encontrados na literatura. Desta forma, discutimos cinco problemas (Seção 2.1) recorrentes em aplicações robóticas: planejamento de caminho (Seção 2.2), localização (Seção 2.3), percepção (Seção 2.4), mapeamento (Seção 2.5) e SLAM (Seção 2.6).

2.1 Problemas fundamentais da robótica

Mover um robô envolve a mudança de estados, por exemplo, da configuração q_{init} para q_{goal} . O estado do robô pode incluir diversas variáveis, como a pose, velocidade e dados dos sensores. A pose compreende a localização e orientação do robô em relação a um sistema de coordenadas global. Para um robô móvel planar não holonômico (que será o robô padrão do presente trabalho), a pose $x_t = (x, y, \theta)^T$ é dada por três variáveis: coordenadas do robô no plano (x e y) e a sua orientação θ . O conjunto de poses válidas para o robô atingir no ambiente é conhecido como configurações livres C_{free} . Já as poses que não são válidas são denominadas regiões proibidas ou obstáculos [Thrun et al., 2005]. Dadas as definições acima, Dudek and Jenkin [2010] estabelecem cinco problemas clássicos da robótica, que serão explanados com mais detalhes nas próximas seções.

Problema 2.1. Planejamento de caminho: É possível levar o robô de uma configuração q_{init} para outra q_{goal} permanecendo dentro de C_{free} ?

Problema 2.2. Localização: Como o robô pode determinar seu estado se possui medições locais de C_{free} ?

Problema 2.3. Percepção: Como o robô pode determinar quais partes do seu ambiente estão ocupadas? Ou seja, como determinar C_{free} ?

Problema 2.4. *Mapeamento*: Como o robô pode determinar C_{free} , assumindo que ele sempre sabe onde está?

Problema 2.5. *SLAM (Mapeamento e Localização Simultâneos)*: Como o robô pode determinar sua pose e C_{free} se elas não são conhecidas?

2.2 Planejamento de caminho

O planejamento de caminho envolve determinar um caminho pertencente ao conjunto C_{free} , que conecta as configurações q_{init} e q_{goal} . É desejável que o caminho seja uma curva contínua em C_{free} , como mostra a Figura 2.1. Existem vários critérios para determinar um caminho, como o que possui menor distância, ou que levará menos tempo para ser percorrido.

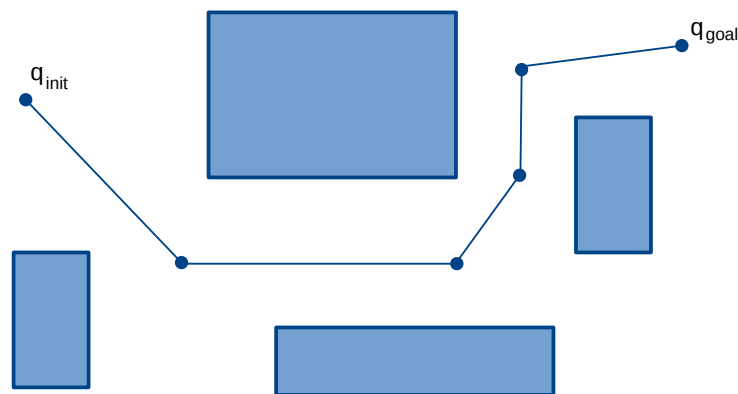


Figura 2.1 – Planejamento do caminho para conectar a configuração q_{init} à q_{goal} .

Na literatura existem vários algoritmos para geração de caminhos para um robô móvel. Portanto, o roboticista deve escolher o que mais se adequa à aplicação a ser desenvolvida. Quatro algoritmos de geração de caminhos (*Bug*, *Dijkstra*, *PRM*, *RRT*) são exibidos abaixo.

- *Bug* (*Bug 0*, *Bug 1*, *Bug 2* e *Tangent Bug*): Encontram um caminho entre q_{init} e q_{goal} em um ambiente desconhecido usando dados sensoriais. Estes algoritmos consideram que o robô possui sensores de contato para detectar obstáculos. Inicialmente o robô deve prosseguir em direção ao alvo, quando ele encontra um obstáculo deve contorná-lo até poder mover na direção do alvo novamente. Este processo deve continuar até que o robô atinja o alvo. O caminho encontrado será uma sequência de pares *hit* (ponto que o robô “toca” no obstáculo ao encontrá-lo) e *leave* (ponto que o robô deixa de seguir a borda do obstáculo) [Choset et al., 2005]. Os pares *hit/leave* são conhecidos como *waypoints*.
- *Dijkstra*: Encontra o caminho ótimo entre q_{init} e q_{goal} de acordo com algum critério de otimalidade (distância, tempo, energia e etc). Para tal, o ambiente é representado por uma estrutura de grafo (ou *grid*), no qual os nós (ou vértices) estão dispostos em C_{free} . O algoritmo de Dijkstra encontra o melhor caminho de q_{init} a todos os outros nós presentes no ambiente, e conseqüente o caminho ótimo até q_{goal} [Cormen et al., 2009].

- *PRM (Probabilistic Roadmap)*: É um método probabilístico que através da geração de configurações livres aleatórias constrói um mapa de rotas. Este mapa é produzido utilizando um planejador local que conecta as configurações geradas. Ao invés de explorar todas as configurações possíveis do ambiente, o PRM explora aleatoriamente um pequeno subconjunto das possibilidades. Uma vez obtido o mapa de rotas, pode-se estabelecer caminhos conectando q_{init} e q_{goal} [Karaman and Frazzoli, 2011].
- *RRT (Rapidly-Exploring Random Tree)*: É um método probabilístico que constrói de forma incremental uma árvore de percursos viáveis, tendo como raiz q_{init} . Inicializa-se o algoritmo RRT com uma árvore possuindo apenas o nó q_{init} . Em cada iteração do RRT, um ponto $x_{rand} \in C_{free}$ é amostrado, e tenta-se conectá-lo ao nó mais próximo da árvore. Se a conexão puder ser estabelecida x_{rand} é adicionado à árvore. A iteração do algoritmo é interrompida no momento que a árvore possuir um nó na região de q_{goal} , produzindo assim um caminho factível entre q_{init} e q_{goal} [Karaman and Frazzoli, 2011].

2.3 Localização

Supondo que o robô está em uma pose $x_t = (x, y, \theta)^T$, e foi executada uma ação u_t que move o robô para uma determinada pose $x_{t+1} = (x + \Delta x, y + \Delta y, \theta + \Delta \theta)^T$, como saber onde o robô está localizado em relação ao mapa do ambiente? Este problema também é conhecido como estimativa da pose ou rastreamento da pose do robô [Dudek and Jenkin, 2010; Thrun et al., 2005]. O modelo gráfico para o problema de localização é mostrado na Figura 2.2. A partir de um mapa do ambiente m fornecido *a priori*, o objetivo é determinar a pose do robô com relação a este mapa, dadas as percepções z e seus controles u . O grande problema é que a maioria dos sistemas robóticos não possuem um sensor sem ruído capaz de medir diretamente a pose. Desta forma, a pose deverá ser inferida a partir das percepções obtidas do ambiente.

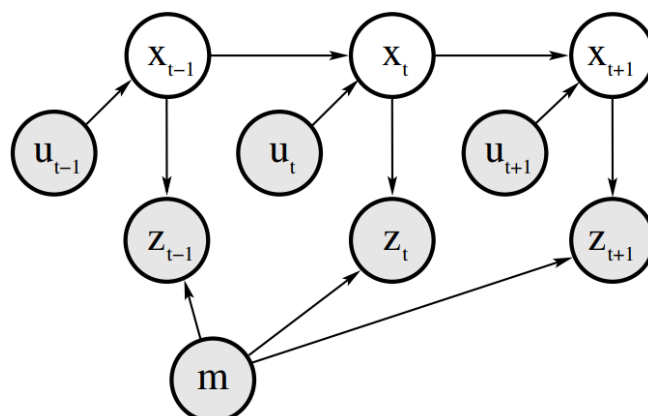


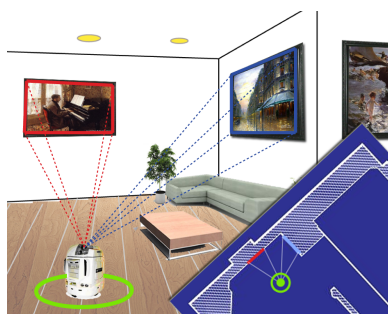
Figura 2.2 – Modelo gráfico para localização de robôs móveis. O objetivo da localização é inferir a verdadeira pose do robô x , a partir das variáveis conhecidas: mapa m , medições z e controles u [Thrun et al., 2005].

Todavia, deve-se estabelecer algum método robusto que seja responsável por estimar a localização do robô durante seu trajeto. A odometria obtida a partir das revoluções das rodas é um dos métodos mais usados na estimação da pose de um robô (existem outros tipos de odometria, como a visual e a laser). A precisão da estimativa da odometria é relativamente boa

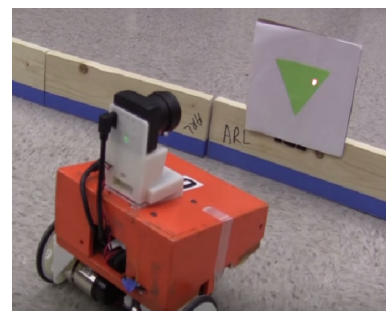
para percursos curtos, já para trajetos maiores a odometria inevitavelmente acumulará erros [Chong and Kleeman, 1997]. As fontes de incertezas da odometria são os erros sistemáticos (desalinhamento das rodas, diâmetro desigual das rodas e etc) e não sistemáticos (acelerações bruscas, forças externas, escorregamento das rodas e etc) [Balbinot and Brusamarello, 2010]. Para se obter uma estimativa de localização robusta a odometria não será suficiente. Logo, ela deve ser usada em conjunto com técnicas mais eficientes.

Uma forma de localizar um robô no mapa é usando informações dos objetos presentes no ambiente. As informações coletadas dos objetos são denominadas *features*. Geralmente, utiliza-se um sistema de visão computacional para realizar a extração de *features*. *Features* comuns de serem coletadas são: linhas, cantos, arestas, cores e etc. Uma *feature* também pode ser uma informação fortemente distinguível no ambiente, como portais, janelas, extintores de incêndio, placas, padrões de fácil reconhecimento e etc. É desejável que as *features* sejam de fácil detecção e identificação. Desta forma, pode-se notar que há inúmeros aspectos do ambiente que podem ser denotados como uma *feature*. Em robótica, costuma-se chamar as *features* utilizadas para navegação e localização de *landmarks*. As *landmarks* geralmente são estáticas e assumem a função de serem pontos de referência no ambiente. Se um robô encontrar uma *landmark* no mundo, e esta aparece em um mapa, conseqüentemente o robô é localizado em relação ao mapa. Uma questão importante é se as *landmarks* são naturais ou artificiais. As artificiais são colocadas no ambiente especificamente para fins de localização, em locais fáceis de serem identificadas, e geralmente são rotuladas como uma assinatura (id). Já as *landmarks* naturais são *features* existentes no ambiente, e são selecionadas para auxiliar na localização (mesmo sem terem sido projetadas para tal). Problemas podem surgir quando as *landmarks* não são rotuladas ou quando o robô encontrar novas *landmarks*, momento em que deverá decidir se elas devem ou não ser adicionadas ao mapa [Dudek and Jenkin, 2010; Murphy, 2000b; Thrun et al., 2005].

Na Figura 2.3 são mostrados exemplos de utilização de *landmarks* na etapa de localização. Na Figura 2.3a temos uma aplicação desenvolvida por Montero et al. [2015], no qual é criada uma estrutura para estimar a localização de robôs móveis a partir de *landmarks* naturais. As *landmarks* a serem capturadas são quadros de pintura. Já a Figura 2.3b mostra uma aplicação desenvolvida por Ghosh [2016], em que a *landmark* é um triângulo verde.



(a) *Landmarks* planares [Montero et al., 2015].



(b) *Landmark* triangular colorida [Ghosh, 2016].

Figura 2.3 – Exemplos de aplicações com robô móvel utilizando *landmarks* para se localizar no ambiente.

Mais detalhes sobre a localização de robôs são vistos no Capítulo 3.

2.4 Percepção

A percepção das informações contidas no ambiente é algo primordial para o desenvolvimento de sistemas robóticos. Para perceber uma situação perigosa à frente, risco de queda ou colisão, o robô deverá fazer uso de um sistema de sensoriamento. Além destas funções, os sensores também podem ser úteis para estimar a pose do robô, construir mapas, medir distâncias percorridas, medir mudanças inerciais e etc [Dudek and Jenkin, 2010; Murphy, 2000b]. Segundo Balbinot and Brusamarello [2010], um sensor é um dispositivo que detecta um estímulo físico ou químico, e fazendo uso de um transdutor pode transformar o sinal detectado (pressão, luz, temperatura, tempo, força e etc) em um sinal analógico que pode ser lido pelo robô. Ou seja, o sensor é o dispositivo que informará ao robô as alterações do mundo e das suas variáveis internas. Portanto, o desafio para o robô é capturar os dados dos sensores e interpretá-los de forma correta, para que possa se localizar e navegar por um ambiente. Exemplos de sensores utilizados na robótica são os proprioceptivos, de posicionamento, de proximidade e visuais. Neste trabalho, daremos destaque principalmente aos sensores visuais.

Com a utilização da visão, os seres humanos conseguem navegar por ambientes complexos, desde que haja uma fonte de iluminação. Assim, é natural estender a visão como um sensor de percepção do ambiente para os robôs móveis, no qual as câmeras serão usadas para criar representações do ambiente de trabalho [Bonin-Font et al., 2008; Dudek and Jenkin, 2010].

Os sistemas robóticos baseados em visão computacional coletam informações do ambiente por meio da energia eletromagnética emitida ou refletida pelos corpos no espectro visível, produzindo uma imagem digital da cena observada. A partir das imagens capturadas, é possível detectar *features* no ambiente, e medir distâncias e velocidades. As duas tecnologias de câmera mais difundidas no mercado são a CCD (do inglês, *Charge Coupled Device*) e CMOS (do inglês, *Complementary Metal Oxide Semiconductor*). As câmeras CCD geralmente têm imagens de maior qualidade e menor ruído. Já as câmeras CMOS têm tamanhos e custos menores, além de consumir menos energia e possuir maior velocidade. Por apresentar melhor qualidade, a tecnologia CCD foi a mais difundida nas últimas décadas, e virou sinônimo de qualidade. Entretanto, nos últimos anos a tecnologia CMOS apresentou avanços na qualidade da imagem, e juntando as vantagens mencionadas acima, o seu desempenho se torna superior ao da CCD [Ballard and Brown, 1982; Dudek and Jenkin, 2010; Lomasney, 2019; Murphy, 2000b]. Pela Figura 2.4, pode-se observar a tendência da câmera CMOS se tornar dominante no mercado. Devido ao aprimoramento das tecnologias de desenvolvimento das câmeras e à queda de preço no decorrer dos anos, a utilização de câmeras nos robôs tem se popularizado cada vez mais.

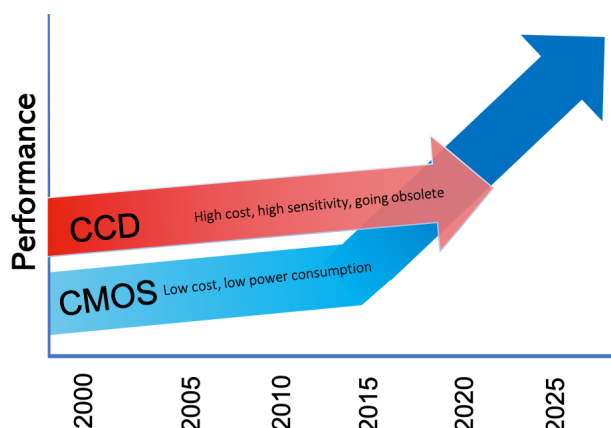
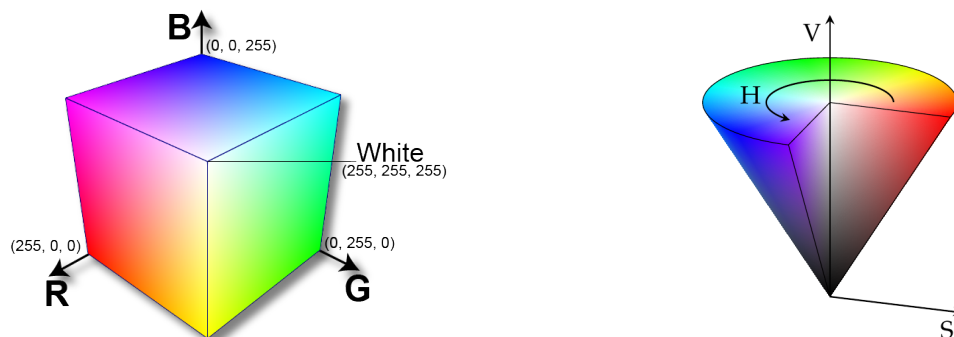


Figura 2.4 – Comparação entre o desempenho das câmeras produzidas com a tecnologia CCD e CMOS, observa-se que nos próximos anos a CMOS será dominante no mercado [Lomasney, 2019].

As imagens capturadas por uma câmera geralmente estão no modelo de cores RGB (do inglês, *Red Green Blue*). Ou seja, a cor de cada *pixel* da imagem é denotada por uma tupla de três valores: vermelho, verde e azul (recebendo um valor de 0 a 255). O modelo de cores RGB pode ser representado por um cubo, como mostra a Figura 2.5a. Uma desvantagem do modelo RGB é não conseguir mensurar a quantidade de luz e brilho presente na imagem. Isto é uma característica ruim, pois os parâmetros de brilho e luz variam de ambiente para ambiente. Uma alternativa é converter a imagem capturada do modelo de cores RGB para o HSV¹ (do inglês, *Hue Saturation Value*) [Ballard and Brown, 1982; Murphy, 2000b].

Na representação HSV cada cor possui uma: intensidade de brilho (*Value*), saturação (*Saturation*) que é representada pela quantidade de luz branca presente na cor, e a matiz (*Hue*) que é a tonalidade da cor predominante. Utilizando estes três parâmetros é possível realizar uma representação espacial, de acordo com a Figura 2.5b. Os valores de intensidade de brilho e saturação variam entre 0 a 100% e os valores da matiz variam de 0° a 360°. Quando um filtro de segmentação de cores HSV for implementado, o mesmo deverá passar por uma etapa de calibração, uma vez que as componentes das cores dependem da incidência de luz do ambiente, e esta variável não é controlada [Souto, 2003].



(a) Representação espacial do modelo de cores RGB (*Red Green Blue*) [MathWorks, 2005]. (b) Representação espacial do modelo de cores HSV (*Hue Saturation Value*) [Duarte, 2015].

Figura 2.5 – Modelo de cores RGB e HSV.

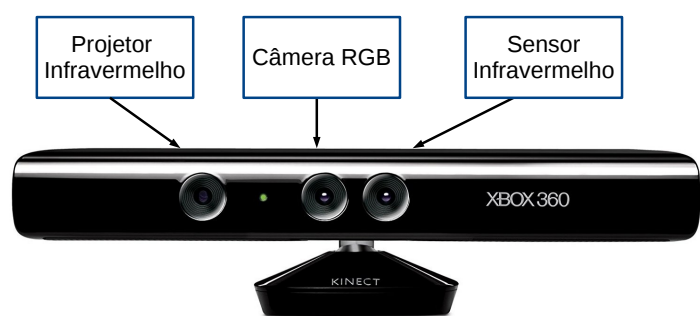
¹ As fórmulas para a conversão do modelo de cores RGB para HSV podem ser encontradas em Chernov et al. [2015].

As câmeras tradicionais possuem apenas uma lente e são conhecidas como monoculares. Porém, extrair a profundidade de uma imagem 2D capturada com uma câmera monocular é uma tarefa complexa. Uma forma de estimar a profundidade é utilizando a visão estéreo, que consiste na combinação de duas ou mais câmeras. O funcionamento da visão estéreo é semelhante ao sensoriamento 3D da visão humana: ela identifica *pixels* que correspondem ao mesmo ponto de uma cena, observados por diferentes câmeras. Uma vez que a distância entre as câmeras é conhecida, pode-se determinar a posição 3D do ponto por meio da triangulação [Flir, 2019; Murphy, 2000b]. A ilustração de um sistema estéreo com duas câmeras é vista na Figura 2.6a.

Outro método de determinar a profundidade dos objetos de uma cena é utilizando as câmeras RGB-D (do inglês, *Red Green Blue - Depth*). As câmeras RGB-D combinam as informações de cores RGB com as de profundidade para cada *pixel* da imagem. Além de uma câmera monocular RGB, a câmera RGB-D possui um módulo emissor de feixes de laser infravermelho e outro para capturar os sinais refletidos quando colidirem com algum objeto da cena. Portanto, os dados de profundidade fornecidos pelo sensor infravermelho serão correlacionados com os da câmeras RGB. Isto implica em uma imagem RGB com profundidade associada a cada *pixel* [Litomisky, 2012]. A câmera RGB-D Kinect da Microsoft pode ser vista na Figura 2.6b, enquanto a Figura 2.7 mostra imagens capturadas com uma câmera RGB-D da Intel RealSense.



(a) Sistema de visão estéreo constituído de duas câmeras [Source, 2019].



(b) Câmera RGB-D Kinect [Microsoft, 2019].

Figura 2.6 – Exemplos de câmeras que extraem informação 3D de uma cena.

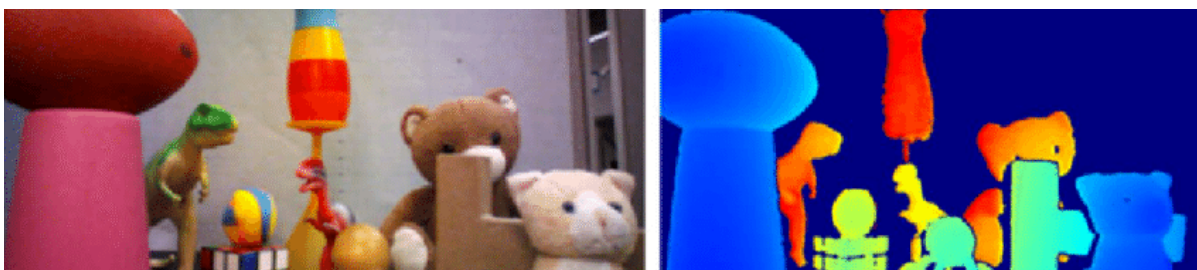


Figura 2.7 – A ilustração da esquerda mostra a imagem de um ambiente capturada com o módulo RGB. Já a ilustração da direita, mostra a imagem de profundidade, na qual os pontos mais próximos são mostrados com cores azuis e os mais distantes com cores vermelhas. Os objetos de plano de fundo, que ultrapassam o alcance do sensor, são representados em azul escuro [Bhowmik, 2017].

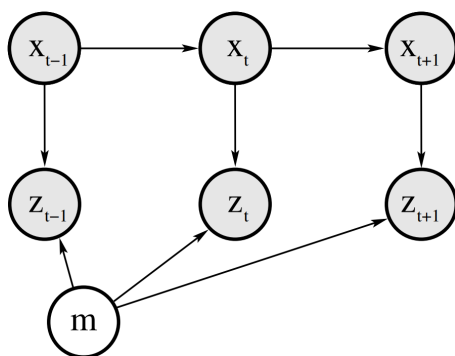
No contexto da robótica móvel, as câmeras de profundidade são utilizadas principalmente para localização e mapeamento 3D.

Nesta seção de percepção, citamos alguns sensores largamente utilizados em aplicações robóticas (dando ênfase aos sensores visuais, pois foram os sensores usados nos experimentos e simulações do presente trabalho). Porém, há de se salientar que os sensores reais são ruidosos e retornam descrições incompletas do ambiente. Logo, deve-se escolher o sensor no qual as especificações sejam adequadas para aplicação a ser desenvolvida. Em muitas situações, um único sensor pode não ser suficiente para estimar uma variável do sistema. Então, utiliza-se a fusão sensorial combinando dados oriundos de vários sensores em uma única percepção. Segundo [Murphy \[2000b\]](#), a combinação dos sensores pode ser: redundante, quando os sensores fornecem as mesmas informações sobre um objeto de interesse; complementar, quando os sensores fornecem informações diferentes sobre um mesmo objeto de interesse; e coordenada, quando os sensores são usados sequencialmente. A grande questão em relação à fusão sensorial é como combinar os dados de diferentes fontes e como dar pesos a estas informações.

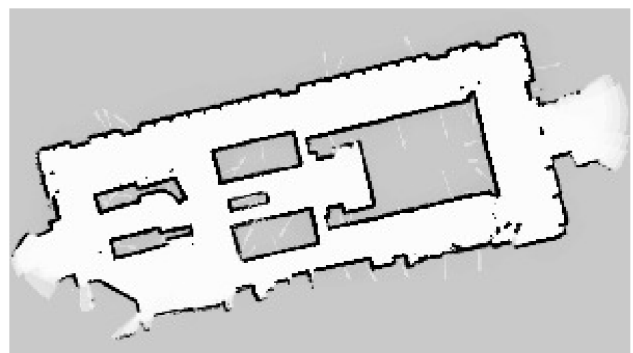
2.5 Mapeamento

Em muitas aplicações de robótica móvel assume-se que o mapa do ambiente é conhecido. Entretanto, são raras as situações que o mapa é fornecido *a priori*. Os projetos arquitetônicos também não são muito confiáveis, pois podem sofrer alterações durante a construção, e podem não conter objetos transitórios como cadeiras, mesas, painéis, *poofs* e etc. Enfim, quase todos os ambientes sofrem mudanças contínuas, e conseqüentemente, os robôs móveis devem ter a capacidade de interpretar as alterações do ambiente e atualizar o mapa [[Dudek and Jenkin, 2010](#)].

Assumindo que as poses do robô durante um percurso são conhecidas, o problema de mapeamento pode ser representado graficamente pela Figura 2.8a. Ou seja, considerando que as variáveis poses x e medições z são conhecidas, como obter o mapa m do ambiente? Partindo da suposição do conhecimento das poses do robô e de suas medições (provenientes de um laser), o mapa do ambiente pode ser construído usando o algoritmo *Occupancy grid map*², como mostra a Figura 2.8b.



(a) Modelo gráfico do mapeamento m com poses x e medições z conhecidas [[Thrun et al., 2005](#)].



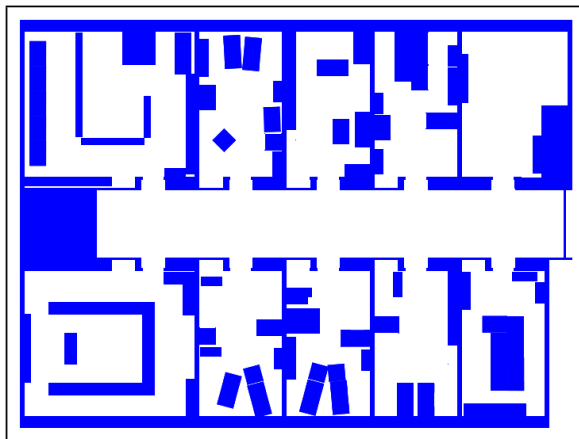
(b) *Occupancy grid map* [[Thrun et al., 2005](#)].

Figura 2.8 – Mapeamento com poses conhecidas.

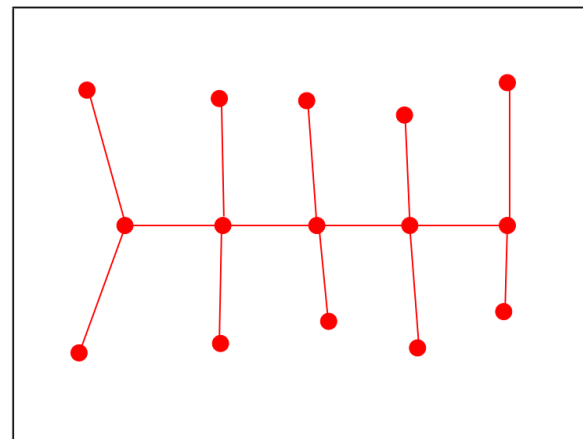
² Mais detalhes sobre este algoritmo podem ser vistos no capítulo 9 do livro de [Thrun et al. \[2005\]](#).

Como a verdadeira pose do robô geralmente não é conhecida, o mapa deverá estar relacionado aos tipos de informações que o robô observa através dos sensores. Desta forma, pode-se criar um mapa utilizando as medições dos sensores juntamente com as informações de odometria. No entanto, os sensores são ruidosos e a odometria acumula erros, criando assim dificuldades para o problema de mapeamento.

Alternativas aos mapas sensoriais são os mapas métricos e topológicos. Os mapas métricos utilizam uma referência absoluta para representar espacialmente as regiões que estão livres e ocupadas no ambiente. Os mapas métricos normalmente são construídos manualmente, o que pode ser uma tarefa inviável para grandes ambientes. Por outro lado, os mapas topológicos são mais simples e extraem as informações de conectividade do ambiente. A representação topológica consiste em reproduzir o espaço de trabalho em grafos formados por nós e arestas. Os nós podem representar locais de interesse no mapa (como portas, janelas, cruzamento de corredores e etc), e as arestas as conexões entre estes locais. Há também a composição híbrida entre os mapas métricos e topológicos, na qual as arestas do mapa topológico contêm informações de distância [Murphy, 2000b]. As Figuras 2.9a e 2.9b mostram o exemplo de um mapa métrico e topológico, respectivamente.



(a) *Layout* de um mapa métrico construído manualmente [Thrun et al., 2005].



(b) Mapa topológico construído em forma de um grafo [Thrun et al., 2005].

Figura 2.9 – Exemplos de mapas utilizados para navegação de robôs.

2.6 SLAM

O problema de Localização e Mapeamento Simultâneos, abreviado por SLAM (do inglês, *Simultaneous Localization and Mapping*), é um dos fundamentais da robótica. Este problema surge quando o robô não conhece sua pose nem tem acesso ao mapa do ambiente. As informações conhecidas são as medições $z_{1:t}$ e controles $u_{1:t}$. O problema de SLAM é mais difícil que os problemas anteriores, pois nele o robô deverá construir o mapa do ambiente enquanto se localiza no mesmo [Dudek and Jenkin, 2010; Murphy, 2000b]. Segundo Thrun et al. [2005], o problema de SLAM pode ser dividido em dois: *online* SLAM e *full* SLAM. Em ambos deseja-se encontrar

o mapa, porém o *online SLAM* encontra somente a pose momentânea do robô x_t , enquanto o *full SLAM* busca encontrar a pose durante todo o percurso $x_{1:t}$. As representações gráficas do *online SLAM* e do *full SLAM* são exibidas na Figura 2.10a e Figura 2.10b, respectivamente.

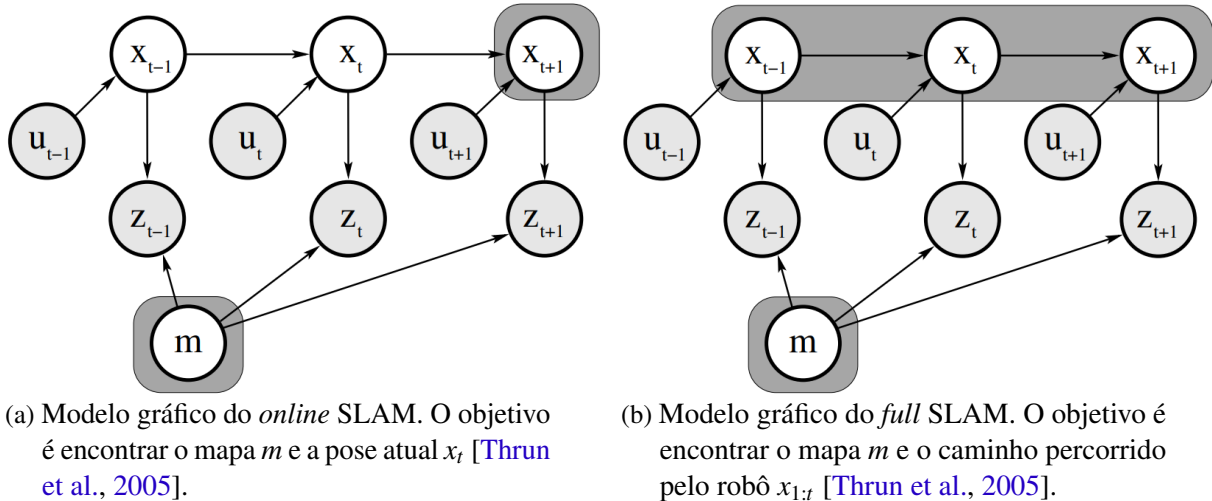


Figura 2.10 – Modelo gráfico do *online SLAM* e *full SLAM*.

No SLAM, os erros locais geralmente são pequenos, mas os erros das estimativas do mapa e da pose tendem a crescer para trajetórias muito longas. Uma forma de corrigir estes erros é usando o *loop closure*. No *loop closure*, o robô se move por um local desconhecido, e quando encontrar uma *feature* já vista, ele detectará que retornou a um local visitado anteriormente e atualizará a pose e o mapa. Um exemplo de *loop closure* é exibido na Figura 2.11. A confirmação do *loop closure* deve ser feita com cautela, pois pode haver um falso positivo. Por exemplo, o robô pode encontrar uma *feature* já vista anteriormente, porém ela pode estar localizada em um local distinto. Assim, a ambiguidade de *features* se torna uma complicação para a confirmação do *loop closure* [Corso, 2013; Thrun et al., 2005].

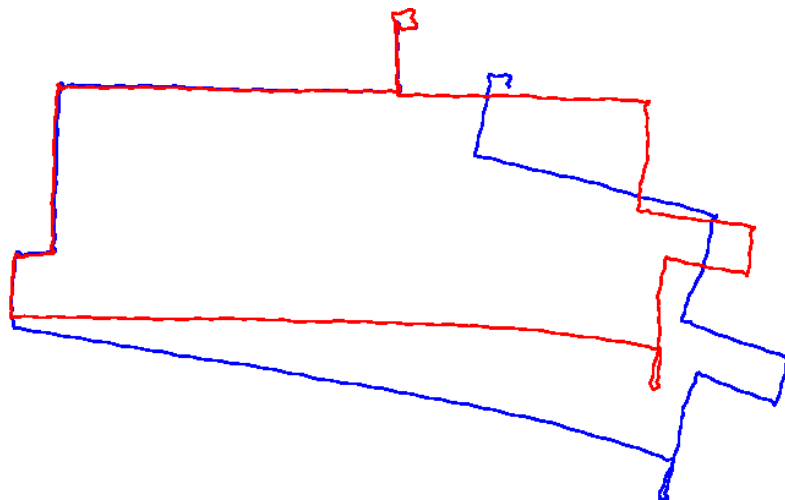


Figura 2.11 – Exemplo de *loop closure*. O trajeto azul indica um caminho típico sem utilizar o *loop closure*. O caminho vermelho mostra o resultado da utilização do *loop closure* após detectar um local já visitado [Corso, 2013].

No próximo capítulo, discutiremos sobre métodos de localização de robôs móveis em ambientes topológicos internos.

LOCALIZAÇÃO DE ROBÔS MÓVEIS TERRESTRES EM AMBIENTES INTERNOS

Os ambientes por onde os robôs se movem podem ser divididos em dois grupos: externos (*outdoors*) e internos (*indoors*). Os ambientes externos também são conhecidos como locais a céu aberto e os internos como locais fechados. Segundo [Liu et al. \[2019\]](#) e [Jamali et al. \[2017\]](#), as pessoas vivem mais de 90% de suas vidas em ambientes internos. Exemplos de ambientes internos são: casas, prédios, salas de aeroportos, supermercados, bibliotecas, escolas, lojas, bancos, restaurantes, indústrias, armazéns e etc.

Os robôs móveis de serviço são desenvolvidos para auxiliarem os seres humanos principalmente em ambientes internos. Algumas tarefas úteis realizadas pelos robôs de serviço em ambientes internos são: limpeza [[Yakoubi and Laskri, 2016](#)], transportar pequenos objetos [[Nieuwenhuisen et al., 2013](#); [Veloso et al., 2015](#)], guiar turistas em museus e exposições [[Al-Wazzan et al., 2016](#); [Thrun et al., 1999](#)], e vigilância [[Correa et al., 2012](#); [Song et al., 2011](#)].

Nos ambientes externos, a pose de um robô móvel pode ser estimada com boa precisão usando o GPS (desde que o local não tenha muitas oclusões). Entretanto, em um ambiente interno o robô não terá acesso às medições do GPS, necessitando assim de outros artifícios para estimar a sua localização. Desta forma, o foco do presente capítulo é fornecer alternativas para estimação da localização de um robô móvel de serviço em ambientes internos, usando *landmarks* artificiais e *features* do próprio ambiente.

Na Seção 3.1, apresentaremos o Modelo Oculto de Markov (MOM) e sua variação ES-MOM, usados para estimar a localização de um robô móvel em um ambiente topológico. Na Seção 3.2, mostraremos a estimação da localização de robôs em espaços de estados contínuos usando o Filtro de Kalman Estendido (FKE). Por fim, exibiremos as simulações e os experimentos na Seção 3.3, e uma breve conclusão do capítulo na Seção 3.4.

3.1 Localização de robôs em ambientes topológicos

Para que o robô navegue de forma precisa pelo ambiente é necessário desenvolver estratégias robustas de localização. Nesta seção, apresentamos um sistema de localização de robôs móveis em um ambiente definido topologicamente³, usando o Modelo Oculto de Markov (do inglês, *Hidden Markov Models*), abreviado pela sigla MOM. Segundo Gomez et al. [2017], o MOM é um método probabilístico e faz uso de uma estrutura perceptiva (a partir de dados de sensores) para obter informações relativas à pose do robô. O MOM fornece uma solução estocástica aplicável à representações discretas, como eventos associados à ações sensoriais. A seguir, definiremos a cadeia de Markov (modelo de Markov mais simples), e na sequência mostraremos uma explanação sobre o MOM, destacando alguns trabalhos relacionados e a sua fundamentação teórica.

3.1.1 Cadeia de Markov

Os modelos de Markov são estocásticos e são usados para representar a evolução de um sistema não determinístico, como o ambiente em torno do robô e o seu estado [Peter, 2016]. Os modelos markovianos têm a propriedade de que a distribuição de probabilidade do próximo estado s_{t+1} depende apenas do estado atual s_t e não da sequência de estados que o precederam $s_{1:t-1}$ [Cassandras and Lafortune, 2009]. Assim, os estados anteriores são irrelevantes para predição de estados futuros, isto é,

$$p(q_{t+1} = s_{t+1} | q_t = s_t, \dots, q_0 = s_0) = p(q_{t+1} = s_{t+1} | q_t = s_t). \quad (3.1)$$

Na Equação (3.1), os estados individuais são indicados por $S = \{s_1, s_2, \dots, s_N\}$ e o estado no instante de tempo t por q_t . Existem variações do modelo de Markov que podem ser usadas em diferentes situações, dependendo se o estado é totalmente observável e se o robô pode afetar este estado tomando decisões. O modelo de Markov mais simples é a cadeia de Markov. Ela modela sistemas totalmente observáveis, em que cada estado corresponde a um evento observável, por exemplo, o ambiente em torno de um robô equipado com um sensor de observação perfeito [Peter, 2016]. A cadeia de Markov (de tempo discreto) é definida por [Cassandras and Lafortune, 2009]:

- Um conjunto de estados finitos S , de tamanho N .
- Uma distribuição inicial de probabilidade $\zeta^4 = \{\zeta_i\}$, em que $\zeta_i = p[q_0 = s_i] \forall s_i \in S$, e $\sum_i^N \zeta_i = 1$.
- Probabilidades de transição de estados.

³ Mais detalhes sobre os mapas topológicos podem ser encontrados na Seção 2.5.

⁴ Na literatura, a distribuição inicial de probabilidades é representada pela letra grega π . Porém, π será usada para simbolizar a política no Capítulo 4. Portanto, substituímos a letra π por ζ .

A principal característica da cadeia de Markov é seu comportamento estocástico definido pelas probabilidades de transição, que podem ser representadas na forma de uma matriz, denominada matriz de transição Γ [Cassandras and Lafortune, 2009]:

$$\Gamma \equiv [\tau_{ij}(t)] \quad i, j = 1, 2, \dots, N. \quad (3.2)$$

no qual, $\tau_{ij}(t)$ é a probabilidade de se atingir o estado s_j estando no estado s_i no instante t , ou de maneira formal:

$$\tau_{ij}(t) \equiv p[q_{t+1} = s_j | q_t = s_i] \quad i, j = 1, 2, \dots, N. \quad (3.3)$$

Como o termo τ_{ij} é uma probabilidade, temos $0 \leq \tau_{ij} \leq 1$ e $\sum_j \tau_{ij} = 1 \forall i$.

A Figura 3.1 mostra uma representação gráfica de uma cadeia de Markov de três estados $S = \{s_1, s_2, s_3\}$, que possui matriz de transição de estados dada por:

$$\Gamma = \begin{bmatrix} 0 & \tau_{12} & \tau_{13} \\ \tau_{21} & 0 & \tau_{23} \\ 0 & \tau_{32} & \tau_{33} \end{bmatrix}. \quad (3.4)$$

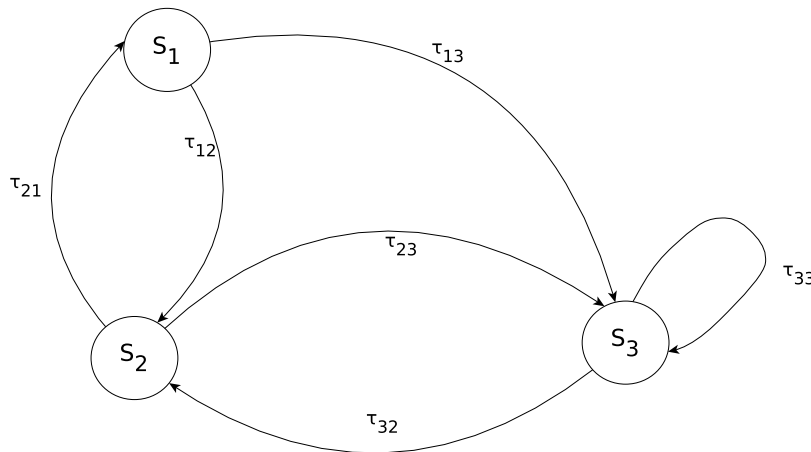


Figura 3.1 – Representação gráfica de uma cadeia de Markov de três estados.

3.1.2 Modelo Oculto de Markov (MOM)

Ao modelar sistemas complexos do mundo real, o conjunto S raramente é observável. Isso geralmente ocorre devido às oclusões naturais ou observações imperfeitas dos sensores [Peter, 2016]. Tais situações podem ser modeladas pelo Modelo Oculto de Markov. O MOM é um modelo de Markov no qual os estados não são diretamente observáveis, ou seja, ocultos. Na cadeia de Markov, considera-se que cada estado corresponde a uma observação, já o MOM representa o cenário no qual a observação é uma função probabilística do estado [Rabiner, 1990].

Um modelo simplificado do MOM é mostrado na Figura 3.2 (um modelo mais completo será exibido na Figura 3.3). Na Figura 3.2, os estados $s_{1:t}$ são ocultos e as únicas informações fornecidas são as observações dos sensores $z_{1:t}$. Assim, uma resolução com MOM pode se

resumir a determinar $p(s_t|z_{1:t})$. Um exemplo de MOM aplicado em robótica é: determinar a localização de um robô em um ambiente definido topologicamente (grafo), em que as observações do espaço de trabalho são provenientes do sistema de percepção do robô. Nesta aplicação, os estados seriam “observados indiretamente” por meio das informações dos sensores $z_{1:t}$.

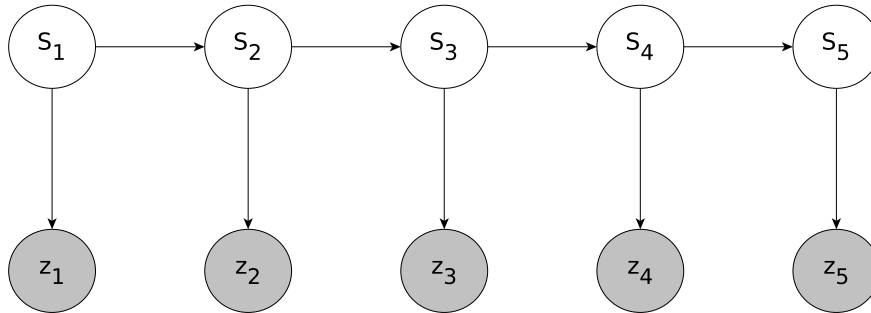


Figura 3.2 – Modelo gráfico simplificado para o MOM [Peter, 2016].

De acordo com Rabiner [1990], um MOM pode ser caracterizado por:

- Número de estados N do modelo. Os estados são denotados por $S = \{s_1, s_2, \dots, s_N\}$.
- Número de símbolos de observação distintos M . Os símbolos de observação correspondem às saídas físicas do sistema que está sendo modelado. Os símbolos individuais são denotados por $L = \{l_1, l_2, \dots, l_M\}$.
- Probabilidades de transição de estados $\Gamma = \{\tau_{ij}\}$, como definido na Seção 3.1.1.
- Distribuição de probabilidade $B = \{b_j(l_k)\}$ do símbolo de observação l_k no estado s_j :

$$b_j(l_k) = p[l_k \text{ em } t | q_t = s_j], \quad j = 1, 2, \dots, N, \quad k = 1, 2, \dots, M, \quad \sum_k b_j(l_k) = 1 \quad \forall j. \quad (3.5)$$

- Distribuição inicial dos estados $\zeta = \{\zeta_i\}$, como definido na Seção 3.1.1.

Acima, podemos ver que a especificação completa de um MOM requer os parâmetros do modelo (N e M), os símbolos de observação L , e as probabilidades Γ, B, ζ . Por conveniência, uma representação compacta do MOM é mostrada na Equação (3.6).

$$\lambda = \{\Gamma, B, \zeta\}. \quad (3.6)$$

Desse modo, o MOM será simbolizado pela tupla λ . A expansão das matrizes ζ, Γ, B , podem ser vistas nas Equações (3.7), (3.8) e (3.9), respectivamente. Nas situações que o estado inicial s_i é bem definido, a probabilidade inicial ζ_i recebe valor um, enquanto as demais recebem zero. Quando o estado inicial é desconhecido, utiliza-se uma distribuição uniforme $\forall \zeta_i$. No caso que o estado inicial for parcialmente conhecido, pode-se usar uma distribuição uniforme para os estados próximos desta localização, e para os demais atribuir valor zero [Thrun et al., 2005]. Na Equação (3.9), as colunas correspondem às observações diferentes e as linhas aos estados. Assim, cada elemento da matriz B indica a probabilidade de se perceber uma observação em determinado estado. No MOM, também temos as observações $O = o_1, o_2, \dots, o_T$, no qual, cada

observação o_t é um dos símbolos possíveis de $L = \{l_1, l_2, \dots, l_M\}$. E T é o número de observações obtidas em sequência. Portanto, a probabilidade de observação também é representada por $b_j(o_t)$, indicando a probabilidade de se observar o_t no estado s_j .

$$\zeta = (\zeta_1 \ \zeta_2 \ \dots \ \zeta_n). \quad (3.7)$$

$$\Gamma = \begin{pmatrix} \tau_{11} & \tau_{12} & \dots & \tau_{1N} \\ \tau_{21} & \tau_{22} & \dots & \tau_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{N1} & \tau_{N2} & \dots & \tau_{NN} \end{pmatrix}. \quad (3.8)$$

$$B = \begin{pmatrix} b_1(l_1) & b_1(l_2) & \dots & b_1(l_M) \\ b_2(l_1) & b_2(l_2) & \dots & b_2(l_M) \\ \vdots & \vdots & \ddots & \vdots \\ b_N(l_1) & b_N(l_2) & \dots & b_N(l_M) \end{pmatrix}. \quad (3.9)$$

Dadas as definições acima, a Figura 3.3 exibe um modelo gráfico detalhado para o MOM.

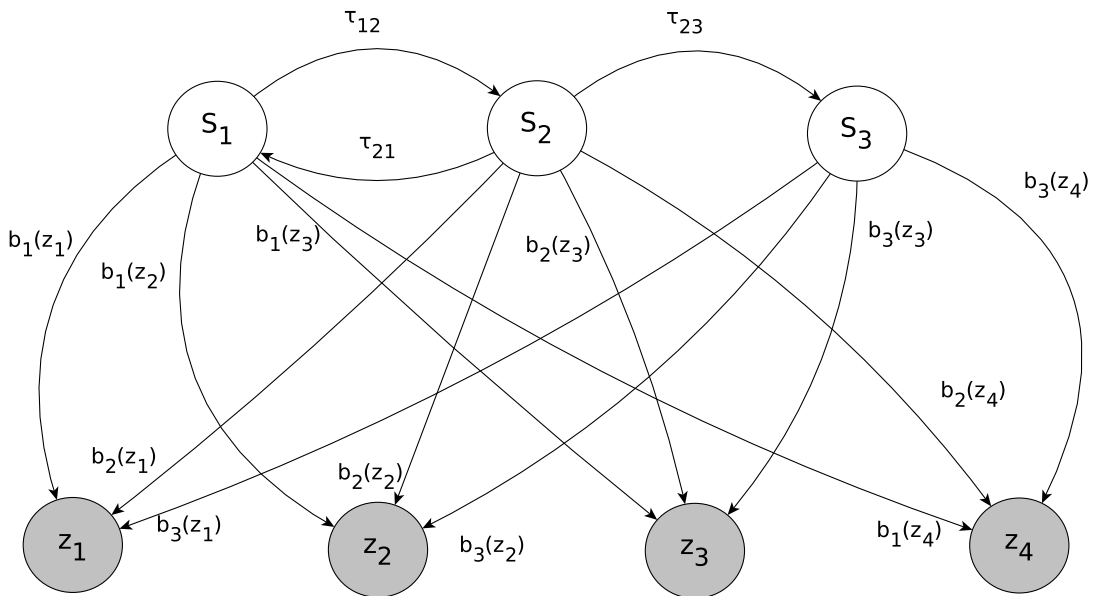


Figura 3.3 – Modelo gráfico detalhado para o MOM [Yin and Meng, 2009].

A seguir, apresentamos um levantamento de trabalhos que usam a estrutura do MOM.

3.1.3 MOM: trabalhos relacionados

Os MOM's podem ser usados em aplicações de navegação e localização de robôs, como pode ser observado na sequência. Rahman and Zelinsky [1999] implementaram um método de navegação de robôs móveis usando MOM, no qual o robô se localiza em um ambiente previamente aprendido. Os dados de um laser foram utilizados para escanear o ambiente e distinguir diferentes locais. O sistema gera uma rede representando uma sequência de locais

associados a uma missão. [Muslim and Ishikawa \[2008\]](#) propuseram uma abordagem baseada em MOM para reconhecer o ambiente de um robô móvel, isto é, a partir de uma sequência de rótulos (observações) o mapa do ambiente é gerado.

[Abreu \[2014\]](#) demonstra a solidez do MOM para resolver o problema da localização de robôs em um ambiente topológico, inclusive para situações de sequestro e de rastreamento de posição. [Gomez et al. \[2016, 2017\]](#) apresentaram um sistema capaz de estimar a localização de um robô móvel em um ambiente definido topologicamente usando o MOM. O sistema de localização desenvolvido requer: uma representação *a priori* do ambiente, a aquisição de percepções relacionadas a eventos, e o planejamento de um caminho. Eles mostraram que a incerteza da localização do robô pode ser consideravelmente diminuída com a aplicação do MOM.

[Nashed et al. \[2018\]](#) apresentaram o MOM de estrutura variável que realiza a estimativa da incerteza da localização topológica, e demonstraram a eficiência da estrutura em um veículo autônomo. [Savage et al. \[2018\]](#) descrevem um sistema de localização para um robô baseado no MOM. Além de ser usado para encontrar a região onde o robô está, o MOM também é utilizado para estimar a orientação do robô. A estimativa da localização é obtida com o algoritmo de Viterbi, a partir das leituras de um laser. [Zhu et al. \[2019\]](#) desenvolveram um método de localização topológica para ambientes internos usando o MOM. Nesta aplicação, eles usam redes neurais para identificar *landmarks* disponíveis no ambiente (escada, extintor, porta, elevador). Por fim, uma série de aplicações usando o MOM podem ser vistas em [Dymarski \[2011\]](#).

3.1.4 Os três problemas do MOM

Com a formulação do MOM, existem três problemas que devem ser solucionados para que este tipo de modelo seja utilizado no mundo real [[Rabiner, 1990](#)]:

Problema 3.1. Dada a sequência de observações $O = o_1, o_2, \dots, o_T$, e um modelo λ , como estimar o estado mais provável do sistema no instante de tempo t ?

Problema 3.2. Dada a sequência de observações $O = o_1, o_2, \dots, o_T$, e um modelo λ , como escolher a sequência de estados correspondente $Q = q_1, q_2, \dots, q_T$ que é ótima em algum sentido?

Problema 3.3. Como os parâmetros do modelo $\lambda = \{\Gamma, B, \zeta\}$ podem ser ajustados para maximizar $p(O|\lambda)$?

O Problema 3.1 resume-se a estimar o estado atual do sistema, uma vez conhecido o modelo λ e as observações O . Este problema é frequentemente resolvido por meio do procedimento *forward-backward* [[Baum and Eagon, 1967](#); [Baum and Sell, 1968](#)]. No Problema 3.2,

deseja-se descobrir a parte oculta do modelo, ou seja, encontrar a sequência “correta” de estados percorrida pelo sistema. Assim, a resolução deste problema é aconselhável para o rastreamento dos estados, uma vez que fornece um histórico de estados mais prováveis, dada uma sequência de observações. Existem várias formas de resolver o Problema 3.2, pois há vários critérios de otimalidade que podem ser escolhidos. O método mais usado para encontrar a melhor solução, dada uma sequência de observações O , é o algoritmo de Viterbi [Forney, 1973; Viterbi, 1967]. No Problema 3.3, tenta-se otimizar os parâmetros do modelo, de modo a descrever melhor como uma dada sequência de observações ocorre. A sequência de observações é usada para ajustar os parâmetros do modelo (“treinar o MOM”) [Baker, 1975; Baum and Sell, 1968].

No presente trabalho, o Problema 3.1 será utilizado para estimar a localização momentânea do robô em um mapa topológico, e o Problema 3.2 para estimar o percurso mais provável percorrido pelo robô neste mapa. Já a aplicação do Problema 3.3 não se encontra no escopo deste trabalho. As principais referências utilizadas para resolução destes problemas foram: Calinon [2017]; Candy [2016]; Cappé et al. [2009]; Elliott et al. [2008]; Rabiner [1990]; Thrun et al. [2005]. Antes de resolver os problemas, definimos alguns conceitos probabilísticos.

Definição 3.1. *Distribuição conjunta:* A distribuição conjunta de duas variáveis aleatórias X e Y é dada por $p(x, y) = p(X = x, Y = y)$, sendo x e y valores específicos que X e Y podem assumir.

Definição 3.2. *Probabilidade condicional:* Suponha que o valor de Y é um y conhecido, e é desejado saber a probabilidade de que X assuma x condicionada a este fato. Tal probabilidade é denotada por $p(x|y) = p(X = x | Y = y)$ e é chamada de probabilidade condicional:

$$p(x|y) = \frac{p(x, y)}{p(y)}. \quad (3.10)$$

Definição 3.3. *Independência condicional:* A combinação de variáveis aleatórias condicionadas na variável z é dada por: $p(x, y|z) = p(x|z)p(y|z)$.

Definição 3.4. *Teorema da probabilidade total:* A partir da definição de probabilidade condicional, pode-se estabelecer o teorema da probabilidade total: $p(x) = \sum_y p(x|y)p(y)$.

Definição 3.5. *Regra de Bayes:* A regra de Bayes relaciona uma probabilidade condicional do tipo $p(x|y)$ à sua inversa $p(y|x)$:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \quad (3.11)$$

Estas definições serão usadas na solução do MOM.

3.1.4.1 Solução do Problema 3.1

No Problema 3.1, deseja-se estimar a probabilidade do sistema estar no estado s_j no instante de tempo t , dadas as observações parciais $o_{1:t} = \{o_1, o_2, \dots, o_t\}$:

$$p(q_t = s_j | o_1, o_2, \dots, o_t) = p(q_t = s_j | o_{1:t}) \stackrel{\text{Eq. (3.10)}}{=} \frac{p(q_t = s_j, o_{1:t})}{p(o_{1:t})} = \frac{\alpha_t(j)}{\sum_{n=1}^N \alpha_t(n)}. \quad (3.12)$$

O termo $\alpha_t(j)$ é conhecido como variável *forward*, e o $\sum_{n=1}^N \alpha_t(n)$ como fator de normalização. A partir da propriedade markoviana, sabemos que o estado s_t depende do estado anterior s_{t-1} , logo, deve-se verificar entre todos os estados do sistema, o mais provável de ser s_{t-1} . Este processo é ilustrado na Figura 3.4. Então, podemos reescrever $\alpha_t(j) = p(q_t = s_j, o_{1:t})$ como:

$$p(q_t = s_j, o_{1:t}) = \sum_{i=1}^N p(q_t = s_j, q_{t-1} = s_i, o_{1:t}) = \sum_{i=1}^N p(q_t = s_j, q_{t-1} = s_i, o_t, o_{1:t-1}). \quad (3.13)$$

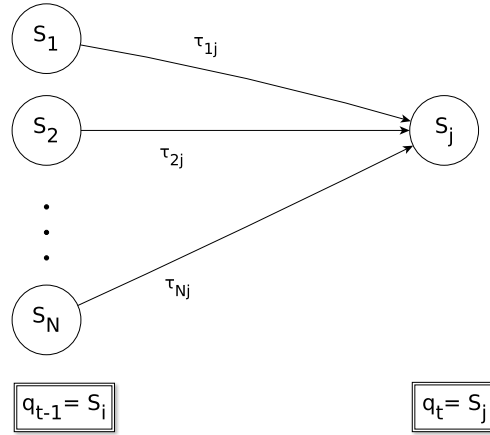


Figura 3.4 – Procedimento *forward*: estado s_t em função do estado s_{t-1} [Rabiner, 1990].

Aplicando a Definição 3.2 na Equação (3.13), obtemos:

$$p(q_t = s_j, o_{1:t}) = \sum_{i=1}^N p(o_t | q_t = s_j, q_{t-1} = s_i, o_{1:t-1}) p(q_t = s_j, q_{t-1} = s_i, o_{1:t-1}). \quad (3.14)$$

$$\begin{aligned} &= \sum_{i=1}^N p(o_t | q_t = s_j, q_{t-1} = s_i, o_{1:t-1}) p(q_t = s_j | q_{t-1} = s_i, o_{1:t-1}) \\ &\times p(q_{t-1} = s_i, o_{1:t-1}). \end{aligned} \quad (3.15)$$

Como a observação o_t é condicionalmente dependente apenas de $q_t = s_j$, podemos reduzir o primeiro termo da Equação (3.15): $p(o_t | q_t = s_j, q_{t-1} = s_i, o_{1:t-1}) = p(o_t | q_t = s_j)$. O segundo termo da Equação (3.15) também pode ser modificado, pois $q_t = s_j$ é condicionalmente dependente apenas do estado anterior: $p(q_t = s_j | q_{t-1} = s_i, o_{1:t-1}) = p(q_t = s_j | q_{t-1} = s_i)$. A partir destas simplificações temos:

$$p(q_t = s_j, o_{1:t}) = p(o_t | q_t = s_j) \sum_{i=1}^N p(q_t = s_j | q_{t-1} = s_i) p(q_{t-1} = s_i, o_{1:t-1}). \quad (3.16)$$

As probabilidades $p(o_t | q_t = s_j)$ e $p(q_t = s_j | q_{t-1} = s_i)$ são equivalentes à probabilidade de observação $b_j(o_t)$ e à probabilidade de transição de estados τ_{ij} , respectivamente. Uma vez que $b_j(o_t)$ e τ_{ij} são modeladas *a priori* no modelo λ , podemos computar $\alpha_t(j) = p(q_t = s_j, o_{1:t})$ de forma recursiva, a partir de $\alpha_{t-1}(i) = p(q_{t-1} = s_i, o_{1:t-1})$. Desta forma, modificamos a Equação (3.16) para que a mesma fique em função de α :

$$\alpha_t(j) = b_j(o_t) \left[\sum_{i=1}^N \tau_{ij} \alpha_{t-1}(i) \right]. \quad (3.17)$$

Com posse de $\alpha_t(j)$ podemos estimar a localização no sistema por meio do Algoritmo 1.

Algoritmo 1: PROCEDIMENTO FORWARD

```

1  início
2  | [1] Inicialização:
3  | Para todo  $j \in N$  faça
4  | |    $\alpha_1(j) = b_j(o_1)\zeta_j$ 
5  | |    $\alpha_1(j) \leftarrow \frac{\alpha_1(j)}{\sum_{n=1}^N \alpha_1(n)}$ 
6  | fim
7  | [2] Indução:
8  | Para  $t = 2$  até  $T$  faça
9  | |   Para todo  $j \in N$  faça
10 | | |    $\alpha_t(j) = b_j(o_t) [\sum_{i=1}^N \tau_{ij} \alpha_{t-1}(i)]$ 
11 | | |    $\alpha_t(j) \leftarrow \frac{\alpha_t(j)}{\sum_{n=1}^N \alpha_t(n)}$ 
12 | | fim
13 | fim
14 fin

```

Na primeira iteração do Algoritmo 1 não temos $\alpha_0(j)$, então substituímos o termo $\sum_{i=1}^N \tau_{ij} \alpha_{t-1}(i)$ por ζ_j na etapa de inicialização. Nas linhas 5 e 11, normalizamos as distribuições de probabilidade $\alpha_1(j)$ e $\alpha_t(j)$, respectivamente. Portanto, obtemos uma solução iterativa para o Problema 3.1, ou seja, uma maneira de calcular $p(q_t = s_j | o_{1:t})$.

Analogamente, podemos considerar a variável *backward* $\beta_t(i)$, outra alternativa de resolver o Problema 3.1. A variável $\beta_t(i)$ determina a probabilidade de observação da sequência parcial (a partir de $t + 1$ até T), dado o estado s_i (no tempo t):

$$\beta_t(i) = p(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i) = p(o_{t+1:T} | q_t = s_i). \quad (3.18)$$

As observações $o_{t+1:T}$ podem ser decompostas em $\{o_{t+1}, o_{t+2:T}, q_{t+1} = s_j\}$. O termo $q_{t+1} = s_j$ foi adicionado, pois a observação o_{t+1} é dada explicitamente em função do estado no instante de tempo $t + 1$. A relação entre $q_{t+1} = s_j$ e $q_t = s_i$ é ilustrada na Figura 3.5, ela mostra que para estar no estado s_i no tempo t (considerando a sequência de observações a partir do tempo $t + 1$), é necessário verificar todos os estados possíveis s_j no tempo $t + 1$. Então, utilizando a Definição 3.2 sobre probabilidade condicional, podemos reescrever $\beta_t(i) = p(o_{t+1:T} | q_t = s_i)$:

$$\beta_t(i) = \sum_{j=1}^N p(o_{t+1}, o_{t+2:T}, q_{t+1} = s_j | q_t = s_i). \quad (3.19)$$

$$= \sum_{j=1}^N \frac{p(o_{t+1}, o_{t+2:T}, q_{t+1} = s_j, q_t = s_i)}{p(q_t = s_i)}. \quad (3.20)$$

$$= \sum_{j=1}^N \frac{p(o_{t+2:T} | o_{t+1}, q_{t+1} = s_j, q_t = s_i) p(o_{t+1}, q_{t+1} = s_j, q_t = s_i)}{p(q_t = s_i)}. \quad (3.21)$$

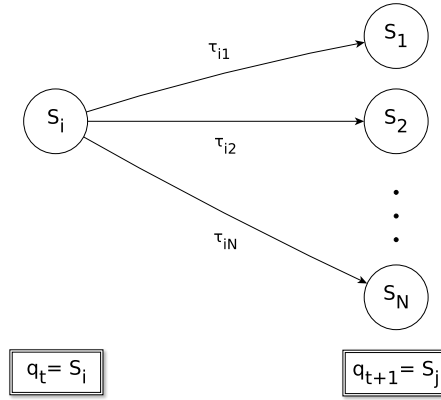


Figura 3.5 – Procedimento *backward*: estado s_t em função do estado s_{t+1} [Rabiner, 1990].

A sequência de observações $o_{t+2:T}$ não é condicionalmente dependente de o_{t+1} nem de $q_t = s_i$, ela dependerá apenas de $q_{t+1} = s_j$, como mostra a Equação (3.18). Assim, o termo $p(o_{t+2:T}|o_{t+1}, q_{t+1} = s_j, q_t = s_i)$ se reduz à probabilidade $p(o_{t+2:T}|q_{t+1} = s_j)$. Utilizando novamente a probabilidade condicional obtemos:

$$\beta_t(i) = \sum_{j=1}^N \frac{p(o_{t+2:T}|q_{t+1} = s_j)p(o_{t+1}|q_{t+1} = s_j, q_t = s_i)p(q_{t+1} = s_j, q_t = s_i)}{p(q_t = s_i)}. \quad (3.22)$$

$$= \sum_{j=1}^N \frac{p(o_{t+2:T}|q_{t+1} = s_j)p(o_{t+1}|q_{t+1} = s_j)p(q_{t+1} = s_j|q_t = s_i)p(q_t = s_i)}{p(q_t = s_i)}. \quad (3.23)$$

$$= \sum_{j=1}^N p(o_{t+2:T}|q_{t+1} = s_j)p(o_{t+1}|q_{t+1} = s_j)p(q_{t+1} = s_j|q_t = s_i). \quad (3.24)$$

Na Equação (3.23), usamos o fato que a observação o_{t+1} não é condicionalmente dependente de $q_t = s_i$, para reduzir $p(o_{t+1}|q_{t+1} = s_j, q_t = s_i)$ à probabilidade $p(o_{t+1}|q_{t+1} = s_j)$. As probabilidades $p(o_{t+1}|q_{t+1} = s_j)$ e $p(q_{t+1} = s_j|q_t = s_i)$ são equivalentes à probabilidade de observação $b_j(o_{t+1})$ e à probabilidade de transição de estados τ_{ij} , respectivamente. Como $b_j(o_t)$ e τ_{ij} são modeladas *a priori* em λ , podemos computar $\beta_t(i) = p(o_{t+1:T}|q_t = s_i)$ de forma recursiva, a partir de $\beta_{t+1}(j) = p(o_{t+2:T}|q_{t+1} = s_j)$. Desta forma, modificamos a Equação (3.24) para que a mesma fique em função da variável β :

$$\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j)b_j(o_{t+1})\tau_{ij}. \quad (3.25)$$

Uma vez que derivamos um método para determinar $\beta_t(i)$, podemos estimar a localização do sistema usando o Algoritmo 2. Na etapa de inicialização, $\beta_T(i)$ é arbitrariamente definida como $1 \forall i$. A variável $\beta_t(i)$ (linha 9) mostra que para se ter estado no estado s_i no tempo t , devem ser considerados todos os estados s_j no tempo $t + 1$, de acordo com as probabilidades de transição τ_{ij} , bem como a probabilidade da observação o_{t+1} no estado s_j ($b_j(o_{t+1})$), e $\beta_{t+1}(j)$. A fim de que $\sum_i \beta_t(i) = 1$, normaliza-se a linha 9 do Algoritmo 2.

Algoritmo 2: PROCEDIMENTO BACKWARD

```

1 início
2   [1] Inicialização:
3   Para todo  $i \in N$  faça
4      $\beta_T(i) = 1$ 
5   fim
6   [2] Indução:
7   Para  $t = T - 1$  até 1 faça
8     Para todo  $i \in N$  faça
9        $\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j) b_j(o_{t+1}) \tau_{ij}$ 
10    fim
11  fim
12 fin

```

É relevante dizer que o procedimento *forward* poderá ser realizado *online* e *offline*, pois ele é dado em função da sequência de observações $o_{1:t}$. Por outro lado, o procedimento *backward* só poderá ser utilizado para estimar a localização no MOM de forma *offline*, isto é, quando todas as observações do sistema tiverem sido coletadas, uma vez que ele é encontrado em função de $o_{t+1:T}$. Assim, o procedimento *forward* poderá ser aplicado em tempo real, e o *backward* como pós-processamento. Nas situações que a estimativa da localização no MOM pode ser realizada *a posteriori*, ela poderá ser refinada por meio da combinação das variáveis *forward* e *backward*. Ou seja, dada a sequência de observações (completa) $O = o_{1:T}$, a probabilidade de o_t ser observada no estado s_i é:

$$\gamma_t(i) = p(q_t = s_i | o_{1:T}) = \frac{p(q_t = s_i, o_{1:T})}{p(o_{1:T})} = \frac{p(o_{1:T} | q_t = s_i) p(q_t = s_i)}{\sum_{i=1}^N p(q_t = s_i, o_{1:t}, o_{t+1:T})}. \quad (3.26)$$

$$\stackrel{\text{Def. 3.3}}{=} \frac{p(o_{1:t} | q_t = s_i) p(o_{t+1:T} | q_t = s_i) p(q_t = s_i)}{\sum_{i=1}^N p(o_{t+1:T} | q_t = s_i, o_{1:t}) p(q_t = s_i, o_{1:t})}. \quad (3.27)$$

$$\stackrel{\text{Eq. (3.10)}}{=} \frac{\overbrace{p(q_t = s_i, o_{1:t})}^{\text{Eq. (3.12)}} \overbrace{p(o_{t+1:T} | q_t = s_i)}^{\text{Eq. (3.18)}}}{\sum_{i=1}^N \underbrace{p(o_{t+1:T} | q_t = s_i)}_{\text{Eq. (3.18)}} \underbrace{p(q_t = s_i, o_{1:t})}_{\text{Eq. (3.12)}}}. \quad (3.28)$$

$$= \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)}. \quad (3.29)$$

O denominador da Equação (3.29) é um fator de normalização, que resulta em $\sum_{i=1}^N \gamma_t(i) = 1$. Usando $\gamma_t(i)$, o estado mais provável no instante de tempo t pode ser encontrado:

$$q_t = \underset{1 \leq i \leq N}{\operatorname{argmax}} [\gamma_t(i)], \quad 1 \leq t \leq T. \quad (3.30)$$

3.1.4.2 Solução do Problema 3.2

Existem várias formas de resolver o Problema 3.2 (diferente do Problema 3.1), pois há vários critérios de otimalidade que podem ser escolhidos. Por exemplo, um critério pode ser escolher o estado $q_t = s_j$ que é o mais provável para uma dada observação. Entretanto, este critério pode apresentar sequências de estados não factíveis caso o MOM possua transições com probabilidade zero ($\tau_{ij} = 0$). Para encontrar a melhor sequência de estados (única) $Q = q_1, q_2, \dots, q_T$, para uma dada sequência de observações $O = o_1, o_2, \dots, o_T$, definimos a grandeza:

$$\delta_t(j) = \max_{q_1, q_2, \dots, q_{t-1}} p(q_1, q_2, \dots, q_t = s_j, o_1, o_2, \dots, o_t), \quad (3.31)$$

isto é, $\delta_t(j)$ representa a probabilidade da melhor sequência de estados que leva ao estado s_j no instante t , com as t primeiras observações.

Aplicando a Definição 3.2 na Equação (3.31), e procedendo de forma análoga ao processo do cálculo das variáveis α e β , obtemos:

$$\delta_t(j) = \max_{q_1, q_2, \dots, q_{t-1}} p(q_1, q_2, \dots, q_t = s_j, o_1, o_2, \dots, o_t). \quad (3.32)$$

$$= \max_{q_{1:t-1}} p(q_{1:t-1}, q_t = s_j, o_{1:t-1}, o_t). \quad (3.33)$$

$$= \max_{q_{1:t-1}} p(o_t | q_{1:t-1}, q_t = s_j, o_{1:t-1}) p(q_{1:t-1}, q_t = s_j, o_{1:t-1}). \quad (3.34)$$

$$= \max_{q_{1:t-1}} p(o_t | q_{1:t-1}, q_t = s_j, o_{1:t-1}) p(q_t = s_j | q_{1:t-1}, o_{1:t-1}) p(q_{1:t-1}, o_{1:t-1}). \quad (3.35)$$

$$= \max_{q_{1:t-1}} p(o_t | q_t = s_j) p(q_t = s_j | q_{t-1} = s_i) p(q_{1:t-2}, q_{t-1} = s_i, o_{1:t-1}). \quad (3.36)$$

$$= p(o_t | q_t = s_j) \max_{q_{1:t-1}} [p(q_t = s_j | q_{t-1} = s_i) p(q_{1:t-2}, q_{t-1} = s_i, o_{1:t-1})]. \quad (3.37)$$

$$= p(o_t | q_t = s_j) \max_{q_{t-1}=s_i} \left[p(q_t = s_j | q_{t-1} = s_i) \max_{q_{1:t-2}} p(q_{1:t-2}, q_{t-1} = s_i, o_{1:t-1}) \right]. \quad (3.38)$$

$$= b_j(o_t) \max_i [\tau_{ij} \delta_{t-1}(i)]. \quad (3.39)$$

Assim, usando a Equação (3.39) podemos determinar $\delta_t(j)$ de forma recursiva. Com posse de $\delta_t(j)$, o rastreamento dos estados q_1, q_2, \dots, q_T é encontrado a partir do Algoritmo 3 (Viterbi).

Na linha 10 do algoritmo de Viterbi, temos a probabilidade de terminar no estado s_j no instante de tempo t , seguindo o caminho mais provável. Para obter a sequência de estados, é necessário manter o rastreamento do argumento que maximiza $\delta_t(j)$, para cada t e s_j . O rastreamento é feito usando $\psi_t(j)$, presente na linha 11 do algoritmo de Viterbi, no qual $\psi_t(j)$ registra o índice do estado que maximiza $\delta_t(j)$. Na linha 16, é executada a etapa de terminação e o estado final mais provável é armazenado em q_T^* . Já na linha 19, o caminho mais provável é recuperado na variável q_t^* .

Algoritmo 3: VITERBI

```

1 início
2   [1] Inicialização:
3   Para todo  $j \in N$  faça
4      $\delta_1(j) = \zeta_j b_j(o_1)$ 
5      $\psi_1(j) = 0.$ 
6   fim
7   [2] Recursão:
8   Para  $t = 2$  até  $T$  faça
9     Para todo  $j \in N$  faça
10       $\delta_t(j) = b_j(o_t) \max_{1 \leq i \leq N} [\delta_{t-1}(i) \tau_{ij}]$ 
11       $\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) \tau_{ij}].$ 
12    fim
13  fim
14  [3] Terminação:
15   $P^* = \max_{1 \leq j \leq N} [\delta_T(j)]$ 
16   $q_T^* = \operatorname{argmax}_{1 \leq j \leq N} [\delta_T(j)]$ 
17  [4] Recriando o caminho (sequência de estados):
18  Para  $t = T - 1$  até 1 faça
19     $q_t^* = \psi_{t+1}(q_{t+1}^*)$ 
20  fim
21 fin

```

3.1.5 ES-MOM

Até o momento, o MOM é constituído basicamente pelas probabilidades de transição de estados e observação. Uma adição ao MOM é assumir que a transição de estados e a observabilidade dependem das ações de entradas $A = a_1, a_2, \dots, a_T$, como pode ser visto na Figura 3.6.

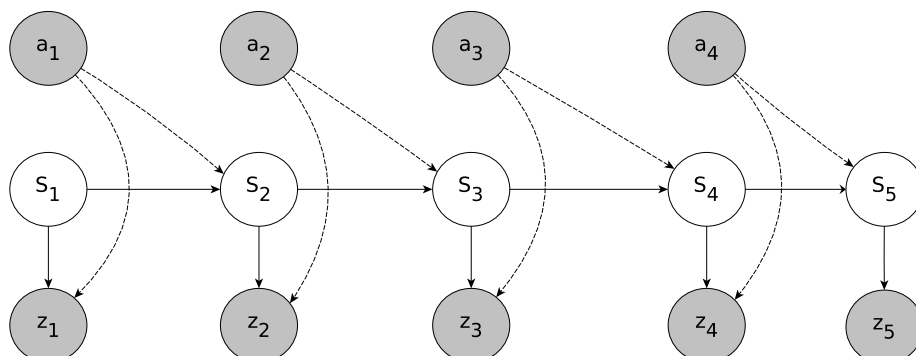


Figura 3.6 – Modelo gráfico simplificado para o ES-MOM [Françoise et al., 2011].

Esta nova formulação do MOM é denominada *Entrada Saída - Modelo Oculto de Markov* (ES-MOM), pois realiza o mapeamento das ações de entrada A nas saídas [Bengio and Frasconi, 1996; Françoise et al., 2011]. Os algoritmos anteriores (*forward*, *backward* e Viterbi) manterão a mesma estrutura, porém as probabilidades de transição e observabilidade, e as variáveis α , β e δ serão modificadas, de acordo com as Equações (3.40) a (3.44), respectivamente.

$$p(q_t = s_j | q_{t-1} = s_i, a_{t-1}). \quad (3.40)$$

$$p(o_t | q_t = s_j, a_{t-1}) = p(o_t | q_t = s_j). \quad (3.41)$$

$$\alpha_t(j) = p(q_t = s_j, o_{1:t} | a_{1:t-1}). \quad (3.42)$$

$$\beta_t(i) = p(o_{t+1:T} | q_t = s_i, a_{1:t-1}). \quad (3.43)$$

$$\delta_t(j) = \max_{q_{1:t-1}} p(q_{1:t-1}, q_t = s_j, o_{1:t} | a_{1:t-1}). \quad (3.44)$$

Na maioria das aplicações, o estado s_j é suficiente para prever a observação o_t , assim a ação a_{t-1} será irrelevante, como mostra a Equação (3.41). Quando a Equação (3.42) é normalizada, ela fornece a probabilidade de estar no estado s_j no instante de tempo t , dadas as observações parciais $o_{1:t}$ e controles $a_{1:t-1}$:

$$p(q_t = s_j | o_{1:t}, a_{1:t-1}) = p(o_t | q_t = s_j) \sum_{i=1}^N p(q_t = s_j | q_{t-1} = s_i, a_{t-1}) p(q_{t-1} = s_i | o_{1:t-1}, a_{1:t-2}). \quad (3.45)$$

A Equação (3.45) é a implementação discreta do filtro de Bayes, e a expressão $p(q_t = s_j | o_{1:t}, a_{1:t-1})$ é conhecida como crença sobre os estados. O filtro de Bayes e a crença serão discutidos com mais detalhes na Seção 3.2.

3.1.6 Resumo da Seção 3.1

Na Seção 3.1 descrevemos o MOM para tempo, observações e estados discretos. O MOM pode se resumir a solução dos três problemas apresentados na Seção 3.1.4. No contexto de navegação robótica, o Problema 3.1 permite localizar em qual estado o robô está, isto é, dada uma sequência de movimentos, o MOM avalia a localização mais provável para o robô, a partir do modelo λ e da sequência de observações O . O Problema 3.2 permite avaliar o trajeto mais provável do robô no espaço de estados até o instante T , ou seja, encontrar a sequência de estados que melhor explica uma sequência de movimentos, a partir do modelo λ e da sequência de observações O .

3.2 Localização de robôs em espaços contínuos

Nesta seção, focaremos nos métodos probabilísticos para localização de robôs móveis em espaços de estados contínuos. Para tal, utilizaremos as mesmas ideias da Seção 3.1, porém aplicadas em um ambiente contínuo. Abaixo apresentamos algumas definições que serão adotadas na presente seção:

- x_t : pose (estado) do robô no instante de tempo t (o estado discreto era representado por s_t).
- z_t : medição dos sensores no instante de tempo t .
- u_t : ação de controle que pode alterar o estado do robô no instante de tempo t (a ação no espaço de estados discreto era representada por a_t).
- $p(x_t|x_{t-1}, u_{t-1})$: probabilidade de transição de estados.
- $p(z_t|x_t)$: probabilidade de medição (observabilidade).

No paradigma probabilístico, a estimativa da localização momentânea do robô é representada pela crença (do inglês, *belief*). A crença é descrita por meio de distribuições de probabilidades condicionadas às medições e ações passadas. A crença sobre uma variável de estado x_t é denotada por $bel(x_t)$ [Thrun et al., 2005]:

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t-1}). \quad (3.46)$$

Usando as definições acima, podemos reescrever a Equação (3.45) no domínio contínuo:

$$bel(x_t) = p(z_t|x_t) \underbrace{\int p(x_t|x_{t-1}, u_{t-1}) bel(x_{t-1}) dx_{t-1}}_{\overline{bel}(x_t)}. \quad (3.47)$$

A expressão $\overline{bel}(x_t)$ é conhecida como predição no contexto de filtros probabilísticos. A predição prevê o estado posterior, antes de serem incorporadas as medições no tempo t . A etapa de calcular $bel(x_t)$ a partir de $\overline{bel}(x_t)$ é chamada de correção ou atualização da medição. A Equação (3.47) é a base do algoritmo do Filtro de Bayes, método mais tradicional para calcular crenças a partir das medições e controles. O algoritmo do Filtro de Bayes é mostrado abaixo.

Algoritmo 4: FILTRO DE BAYES

Entrada: $bel(x_{t-1}), u_{t-1}, z_t$

Saída: $bel(x_t)$

1 **início**

2 **Para todo** x_t **faça**

3 $\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) bel(x_{t-1}) dx_{t-1}$

4 $bel(x_t) = p(z_t|x_t) \overline{bel}(x_t)$

5 **fim**

6 **fin**

3.2.1 Filtro de Kalman

O Filtro de Kalman (FK) é um filtro recursivo proposto por Kalman [1960], e é a implementação mais estudada do Filtro de Bayes. Ele estima o estado atual de um sistema dinâmico linear a partir das observações adquiridas até o momento. Ele também pode ser considerado como uma versão do algoritmo *forward* do ES-MOM, no qual as observações e

os estados ocultos são contínuos. Existe também a versão contínua do procedimento *backward*, o Suavizador de Kalman (do inglês, *Kalman Smoother*), que permite suavizar estimativas de estados anteriores, a partir de observações posteriores. No FK, a crença é representada por uma distribuição gaussiana multivariável, parametrizada por uma média μ_t e covariância Σ_t . A média μ é um vetor que tem a mesma dimensão de x , enquanto a covariância Σ é uma matriz quadrada, positiva semidefinida, com mesma dimensionalidade de x . Uma característica importante é que as distribuições gaussianas são unimodais, ou seja, possuem um único máximo, possibilitando então serem usadas para o rastreamento da pose em muitos problemas robóticos [Thrun et al., 2005]. O modelo dinâmico linear da Equação (3.48) descreve a predição do estado usada no FK, e a Equação (3.49) a predição da medição:

$$x_t = \mathbb{A}_t x_{t-1} + \mathbb{B} u_{t-1} + \varepsilon_x, \quad \varepsilon_x \sim \mathcal{N}(0, \mathbb{R}), \quad (3.48)$$

$$\hat{z}_t = \mathbb{C}_t x_t + \varepsilon_z, \quad \varepsilon_z \sim \mathcal{N}(0, \mathbb{Q}). \quad (3.49)$$

Sendo, $x_t \in \mathfrak{R}^{n \times 1}$ e $x_{t-1} \in \mathfrak{R}^{n \times 1}$ os vetores de estado, $u_{t-1} \in \mathfrak{R}^{m \times 1}$ o vetor de controle e $\hat{z}_t \in \mathfrak{R}^{k \times 1}$ o vetor de medição. \mathbb{A}_t e \mathbb{B}_t são matrizes $\mathfrak{R}^{n \times n}$ e $\mathfrak{R}^{n \times m}$, respectivamente, e descrevem a evolução temporal do estado. Os vetores gaussianos aleatórios $\varepsilon_x \in \mathfrak{R}^{n \times 1}$ e $\varepsilon_z \in \mathfrak{R}^{k \times 1}$, possuem média zero e covariância denotada por \mathbb{R} e \mathbb{Q} , respectivamente, modelando as incertezas introduzidas pela transição de estados e pela medição. A matriz $\mathbb{C}_t \in \mathfrak{R}^{k \times n}$ realiza o mapeamento entre x_t e \hat{z}_t , no qual k é a dimensão do vetor de medição \hat{z}_t [Thrun et al., 2005]. Assim, o FK estima (de forma ótima) o estado de um sistema dinâmico linear, a partir de medições ruidosas. Para gerar uma estimativa dos estados, o FK utiliza o modelo do sistema, as entradas de controle e as medições. O FK é mostrado no Algoritmo 5.

Algoritmo 5: FILTRO DE KALMAN

Entrada: $\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$

Saída: μ_t, Σ_t

1 **início**

2 *Predição:*

3 $\bar{\mu}_t = \mathbb{A}_t \mu_{t-1} + \mathbb{B} u_{t-1}$

4 $\bar{\Sigma}_t = \mathbb{A}_t \Sigma_{t-1} \mathbb{A}_t^T + \mathbb{R}_t$

5 *Atualização:*

6 $K_t = \bar{\Sigma}_t \mathbb{C}_t^T (\mathbb{C}_t \bar{\Sigma}_t \mathbb{C}_t^T + \mathbb{Q}_t)^{-1}$

7 $\mu_t = \bar{\mu}_t + K_t (z_t - \mathbb{C}_t \bar{\mu}_t)$

8 $\Sigma_t = (I - K_t \mathbb{C}_t) \bar{\Sigma}_t$

9 **fim**

A dedução completa do algoritmo do Filtro de Kalman pode ser vista em Briers et al. [2010]; Fraser [2008]; Russell and Norvig [2016]; Thrun et al. [2005]. A entrada do FK é a crença no tempo $t - 1$, representada por μ_{t-1} e Σ_{t-1} . Para atualizar estes parâmetros utiliza-se a entrada de controle u_{t-1} e a medição z_t . A saída do FK é a crença no tempo t , representada por μ_t e Σ_t . A

variável K_t representa ganho de Kalman, e especifica o grau em que a medida z_t é incorporada na nova estimativa do estado. Na linha 7, manipula-se a média $\bar{\mu}_t$, ajustando-a proporcionalmente ao ganho K_t e ao desvio da atual medida z_t . Por fim, calcula-se na linha 8 a nova covariância da crença. Os ruídos do processo e da medição são assumidos como independentes e normalmente distribuídos com covariâncias \mathbb{R}_t e \mathbb{Q}_t , respectivamente.

Em suma, o FK combina uma predição do estado de um sistema dinâmico linear com uma nova medida usando média ponderada (atualização). O resultado da média é uma nova estimativa do estado, que se localiza entre o estado predito e o medido, apresentando uma melhor incerteza estimada que qualquer um dos dois unicamente. O processo é repetido a cada iteração, com a nova estimativa e sua covariância sendo usadas na predição da iteração seguinte.

3.2.2 Filtro de Kalman Estendido

As suposições de linearidade das funções de observabilidade e transitoriedade são cruciais para utilização do FK. Entretanto, estas funções raramente são lineares na prática. O Filtro de Kalman Estendido (FKE) é usado quando o sistema apresenta não linearidade. O FKE lineariza o modelo em torno da estimação atual e em seguida utiliza as equações do FK tradicional. O FKE é baseado na linearização da função de transição de estados e da função de medição por meio da aproximação por série de Taylor, fazendo o truncamento no termo de primeira ordem, e desprezando os termos de ordem superior. Esta aproximação pode causar problemas se as não linearidades são muito grandes, fazendo o algoritmo divergir [Särkkä, 2013; Thrun et al., 2005]. O FKE é mostrado no Algoritmo 6.

Algoritmo 6: FILTRO DE KALMAN ESTENDIDO

Entrada: $\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$

Saída: μ_t, Σ_t

1 **início**

2 | *Predição:*

3 | $\bar{\mu}_t = g(u_{t-1}, \mu_{t-1})$

4 | $\bar{\Sigma}_t = \mathbb{G}_t \Sigma_{t-1} \mathbb{G}_t^T + \mathbb{R}_t$

5 | *Atualização:*

6 | $K_t = \bar{\Sigma}_t \mathbb{H}_t^T (\mathbb{H}_t \bar{\Sigma}_t \mathbb{H}_t^T + \mathbb{Q}_t)^{-1}$

7 | $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$

8 | $\Sigma_t = (I - K_t \mathbb{H}_t) \bar{\Sigma}_t$

9 **fin**

Como pode ser visto no Algoritmo 6, as Equações (3.48) e (3.49) foram relaxadas, e a probabilidade transição de estados e medição são descritas por $x_t = g(u_{t-1}, x_{t-1}) + \varepsilon_x$ e $\hat{z}_t = h(x_t) + \varepsilon_z$, respectivamente, para funções g e h arbitrárias. A ideia do FKE é linearizar as funções g e h usando a aproximação via expansão de Taylor de primeira ordem:

$$g(u_{t-1}, x_{t-1}) \approx g(u_{t-1}, \mu_{t-1}) + \underbrace{\frac{\partial g(u_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}}}_{\mathbb{G}_t} (x_{t-1} - \mu_{t-1}), \quad (3.50)$$

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t}}_{\mathbb{H}_t} (x_t - \bar{\mu}_t). \quad (3.51)$$

As matrizes \mathbb{G}_t e \mathbb{H}_t são conhecidas como Jacobianas de $g(\cdot)$ e $h(\cdot)$, respectivamente.

3.2.2.1 Modelo cinemático do robô diferencial

Neste trabalho, o robô utilizado na maioria dos experimentos e simulações é o diferencial, também conhecido como não holonômico. A Figura 3.7 exibe uma imagem do robô com acionamento diferencial, sendo $\{X_W, Y_W\}$ o *frame* inercial do mundo e $\{X_R, Y_R\}$ o *frame* do robô.

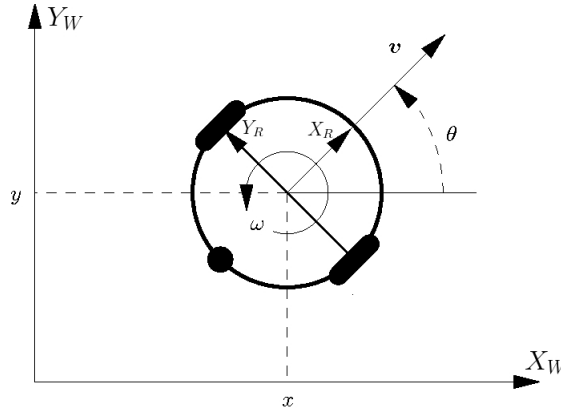


Figura 3.7 – Esboço da vista superior de um robô com acionamento diferencial [Antonelli et al., 2010].

O modelo da pose atual do robô $\mu_t = (x_t, y_t, \theta_t)^T$, após ser executado o comando de movimento $u_{t-1} = (v_{t-1}, \omega_{t-1})^T$ no estado $\mu_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})^T$, é dado por:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}}_{\mu_t} = \underbrace{\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t \cos(\theta_{t-1}) & 0 \\ \Delta t \sin(\theta_{t-1}) & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} v_{t-1} \\ \omega_{t-1} \end{bmatrix}}_{g(u_{t-1}, \mu_{t-1})} + \underbrace{\mathcal{N}(0, \mathbb{R})}_{\varepsilon_x}. \quad (3.52)$$

O parâmetro Δ_t é o período de amostragem, v_{t-1} a velocidade linear, ω_{t-1} a velocidade angular, e μ_t é a estimativa da pose do robô. Pode-se observar na Equação (3.52) que $g(u_{t-1}, \mu_{t-1})$ é claramente não linear, logo o FKE deverá ser utilizado para estimativa da pose. A matriz \mathbb{G}_t do Algoritmo 6 é dada por:

$$\mathbb{G}_t = \frac{\partial g(u_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_{t-1}} & \frac{\partial g_1}{\partial y_{t-1}} & \frac{\partial g_1}{\partial \theta_{t-1}} \\ \frac{\partial g_2}{\partial x_{t-1}} & \frac{\partial g_2}{\partial y_{t-1}} & \frac{\partial g_2}{\partial \theta_{t-1}} \\ \frac{\partial g_3}{\partial x_{t-1}} & \frac{\partial g_3}{\partial y_{t-1}} & \frac{\partial g_3}{\partial \theta_{t-1}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t v_t \sin(\theta_{t-1}) \\ 0 & 1 & \Delta t v_t \cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.53)$$

Nota-se agora que a matriz \mathbb{G}_t é linear, pois $\sin(\theta_{t-1})$ e $\cos(\theta_{t-1})$ são valores numéricos. A matriz \mathbb{G}_t não é constante e deve ser atualizada a cada iteração. O mesmo raciocínio de linearização aplicado para obter a Jacobiana \mathbb{G}_t para o robô não holonômico, também é válido para encontrar a Jacobiana de $h(\bar{\mu}_t)$.

3.3 Simulações e Experimentos

3.3.1 Plataforma robótica

Nos experimentos reais deste trabalho utilizamos a plataforma robótica mostrada na Figura 3.8, composta por uma base móvel não holonômica iRobot Create, um computador Intel Core i7-7500U CPU 2.7GHz x 4 com sistema operacional Ubuntu 16.04, duas câmeras monoculares RGB, uma câmera RGB-D Intel RealSense D435 e uma câmera Intel RealSense Tracking T265 (cedida pelo Instituto Tecnológico da Vale (ITV)). Apesar da plataforma ter quatro câmeras, elas não foram usadas concomitantemente em todos os experimentos. O *framework* usado para integrar o *hardware* e o *software* foi o ROS (*Robot Operating System*) *Kinetic*. O ROS é uma coleção de ferramentas, bibliotecas e convenções que auxiliam na construção de aplicações robóticas. Ele possui uma estrutura que padroniza a comunicação com *hardwares* de diferentes fabricantes por meio da estrutura de nós e tópicos. Os nós são programas que podem receber e enviar dados entre si, e estes dados são denominados tópicos. O processamento das imagens capturadas pelas câmeras foi realizado por meio da biblioteca computacional *OpenCV*. O *driver* de comunicação da base móvel com o ROS foi o *create_autonomy*. Os programas desenvolvidos foram feitos em linguagem C++ e Matlab. A seguir, serão apresentadas as simulações e os experimentos reais do capítulo.

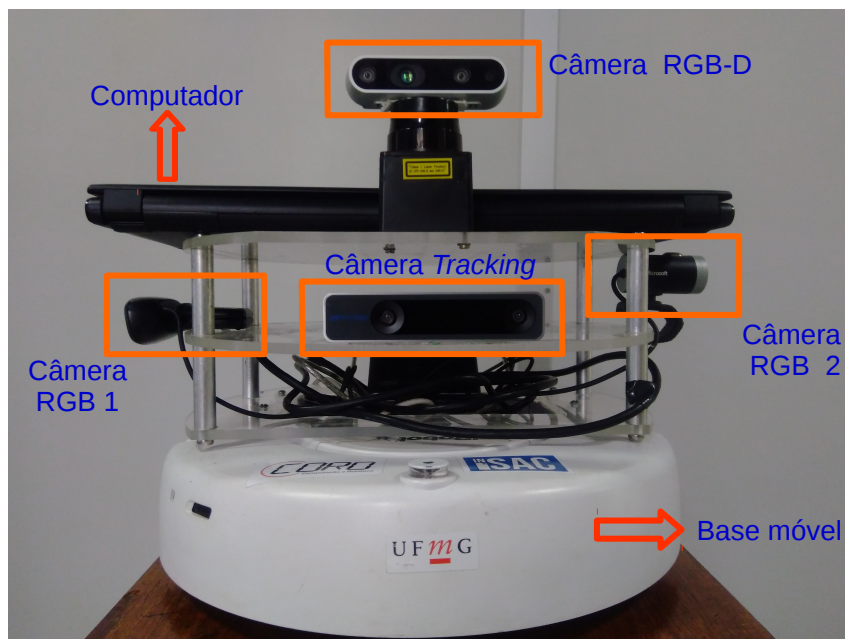


Figura 3.8 – Plataforma robótica utilizada nos experimentos reais.

3.3.2 Experimento com o Filtro de Kalman Estendido

Conforme descrito na Seção 2.3, a odometria não consegue estimar de forma robusta a localização do robô devido ao acúmulo de erros. Desta forma, mostraremos na presente seção alguns experimentos reais utilizando o FKE para estimar a localização do sistema robótico da Figura 3.8 em corredores. Antes de relatar os experimentos, vamos descrever as medições z obtidas com as câmeras RGB-D, *Tracking* e as RGB's, usadas na fase de atualização do FKE.

A câmera RGB-D foi acoplada na frente do robô, e é utilizada para estimar a pose dele por meio do **ORB-SLAM 2**. O ORB-SLAM 2 é um pacote do ROS desenvolvido por **Mur-Artal and Tardós [2017]**, e implementa o SLAM em tempo real. Este pacote é capaz de realizar o rastreamento da câmera RGB-D acoplada no robô, além de construir o mapa 3D do ambiente (a partir das *features* capturadas), e detectar o *loop closure*. O ORB-SLAM 2 tem a opção de ser usado no modo SLAM e no modo localização. O modo localização pode ser usado quando o mapa de *features* do ambiente é fornecido *a priori*. Neste modo, o mapeamento e o *loop closure* são desativados. Como o foco do capítulo está na parte de localização, utilizaremos o pacote ORB-SLAM 2 no modo SLAM, porém só faremos uso da informação da pose e ignoraremos o mapa do ambiente. Portanto, a partir da leitura da imagem RGB e das informações de profundidade, o ORB-SLAM 2 fornece um tópico do ROS com a informação da pose do robô⁵ em tempo real. As *features* detectadas no ambiente (quadrados verdes) e o mapa de *features* podem ser visualizados usando o **Rviz**, conforme mostra a Figura 3.9.

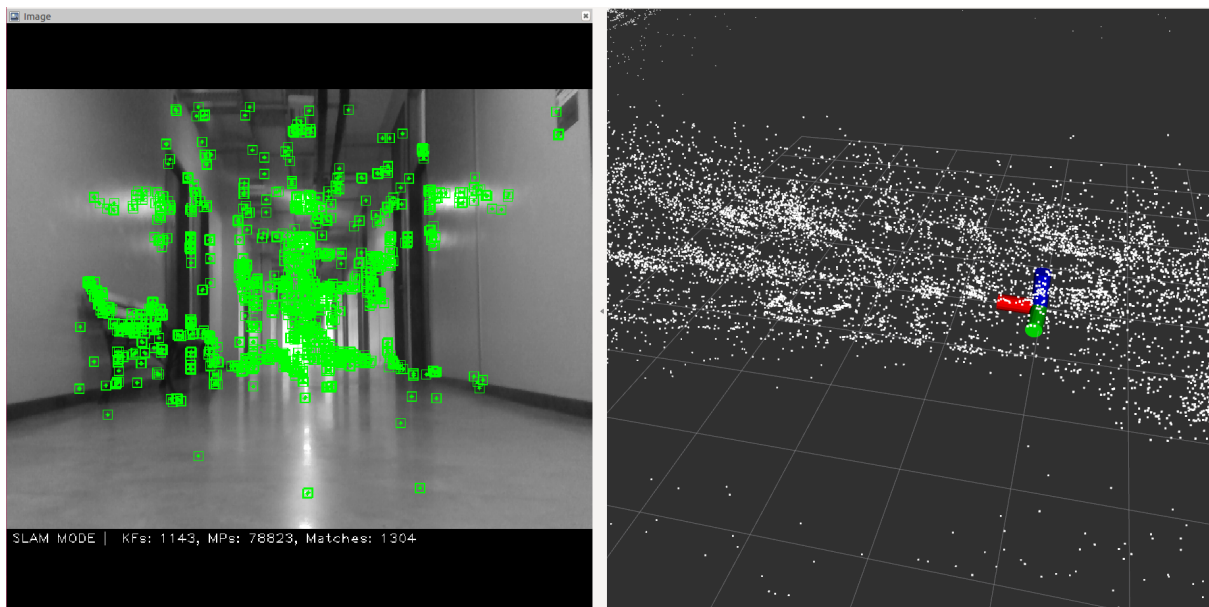


Figura 3.9 – Visualização do funcionamento do ORB-SLAM 2 no Rviz. Na imagem à esquerda, temos as *features* identificadas no ambiente pela câmera RGB-D (destacadas pelos quadrados verdes). Na imagem à direita, temos o mapa de *features* e o sistema de referência da câmera RGB-D.

A câmera *Tracking* também foi colocada na frente do robô, e baseado nas *features* do ambiente implementa o SLAM internamente. Para tal, ela utiliza duas câmeras “olho de peixe” e uma unidade de medida inercial (IMU). A câmera *Tracking* é integrada ao ROS, fornecendo o rastreamento da pose da câmera/robô por meio de um tópico.

Enquanto as câmeras RGB-D e *Tracking* capturam *features* do ambiente (principalmente quinas e locais com grande variação de gradiente na imagem), as câmeras RGB’s (colocadas nas laterais do robô) são utilizadas para identificar *landmarks*. A *landmark* utilizada nos experimentos foi o Aruco (exibido na Figura 3.10), um marcador robusto a variações de luminosidade, pontos

⁵ Para obter a pose do robô é necessário conhecer a transformação do sistema de referência da câmera RGB-D para o sistema de referência do robô.

de vista e escalas nas imagens. O Aruco é denominado um marcador fiducial⁶ planar baseado em quadrados com códigos binários desenvolvido por Garrido-Jurado et al. [2014]. O Aruco possui uma borda preta e uma matriz binária interna que determina seu identificador (id). O tamanho da matriz interna do Aruco da Figura 3.10 é 6x6. Garrido-Jurado et al. [2014] disponibilizaram 250 Arucos diferentes (de tamanho 6x6) para download, cada um possuindo uma id. Para identificar os Arucos utilizamos a *ArUco*, biblioteca *open source* e escrita em linguagem C++.

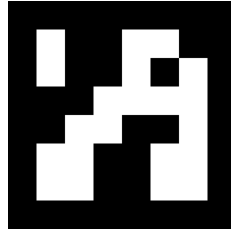


Figura 3.10 – Marcador fiducial Aruco [Garrido-Jurado et al., 2014].

Após a detecção do Aruco, é possível estimar a pose da câmera. Para tal, é necessário obter os *parâmetros internos* de distorção das câmeras monoculares. Estes parâmetros podem ser encontrados por meio da *calibração*. Com posse dos parâmetros internos da câmera e dos parâmetros do Aruco detectado, podemos estimar a transformação do sistema de referência do Aruco em relação ao da câmera através da função *solvePNP*. Esta função retorna o vetor de rotação $rVec = (\theta_x, \theta_y, \theta_z)^T$ e o de translação $tVec = (t_x, t_y, t_z)^T$. As componentes de $tVec$ representam as distâncias em X, Y e Z do Aruco em relação ao sistema de referência da câmera. Já $rVec$ é o vetor de Rodrigues e é convertido para uma matriz de rotação $M_{Aruco}^{C\u00e2mera} \in SO(3)$. Denotaremos a matriz de transformação homogênea $H_B^A \in SE(3)$ [Spong et al., 2006]:

$$H_B^A = \begin{bmatrix} M_B^A & t_{AB}^A \\ 0 & 1 \end{bmatrix}, \quad (3.54)$$

sendo $M_B^A \in SO(3)$ uma matriz de rotação do sistema de referência A para o B , e $t_{AB}^A \in \mathfrak{R}^{3 \times 1}$ um vetor de translação do sistema de referência A para o B , tomando o sistema A como referência.

Para o caso da transformação do sistema de referência do Aruco para câmera, temos a seguinte matriz de transformação homogênea:

$$H_{Aruco}^{C\u00e2mera} = \begin{bmatrix} M_{Aruco}^{C\u00e2mera} & tVec \\ 0 & 1 \end{bmatrix}. \quad (3.55)$$

De forma genérica, podemos definir uma matriz de transformação homogênea entre dois sistemas através das matrizes de transformação “intermediárias”:

$$H_n^0 = H_1^0 H_2^1 \dots H_n^{n-1}. \quad (3.56)$$

Fazendo uso da Equação (3.56), podemos determinar a localização global do robô, assumindo que a localização do Aruco no mundo é conhecida:

$$H_{Rob\u00f4}^{World} = H_{Aruco}^{World} H_{C\u00e2mera}^{Aruco} H_{Rob\u00f4}^{C\u00e2mera}. \quad (3.57)$$

⁶ Um sistema de marcadores fiduciais é composto por um conjunto de marcadores v\u00e1lidos e um algoritmo que realiza sua detec\u00e7\u00e3o em imagens.

As transformações são ilustradas na Figura 3.11. Na Equação (3.57), $H_{\text{Aruco}}^{\text{World}}$ é a transformação do Aruco em relação ao *frame* inercial do mundo, e $H_{\text{Robô}}^{\text{Câmera}}$ é a transformação do *frame* do robô em relação ao *frame* da câmera. Estas transformações serão constantes nos experimentos. Já a transformação $H_{\text{Câmera}}^{\text{Aruco}}$ será calculada toda vez que um Aruco for detectado, e será dada pelo inverso da Equação (3.55): $H_{\text{Câmera}}^{\text{Aruco}} = (H_{\text{Aruco}}^{\text{Câmera}})^{-1}$. Assim, a transformação do robô em relação ao mundo $H_{\text{Robô}}^{\text{World}}$ é obtida. Utilizando a transformação homogênea $H_{\text{Robô}}^{\text{World}}$, a pose do robô será determinada pelas componentes de translação em X e Y , e pela rotação em relação ao eixo Z .

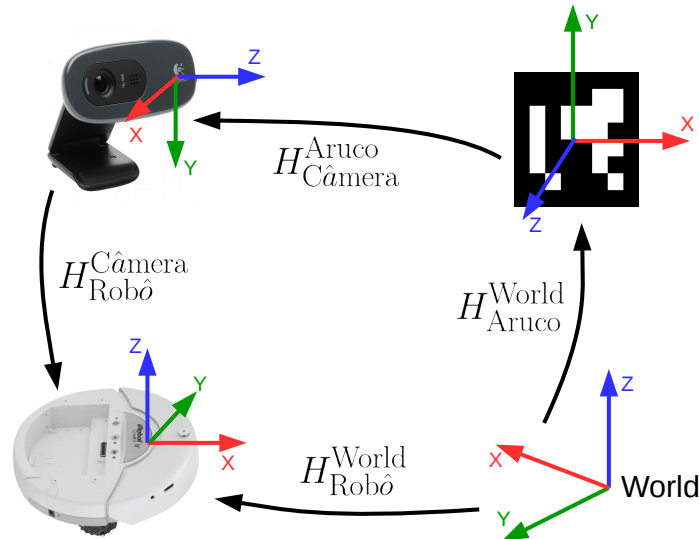


Figura 3.11 – Transformações necessárias para encontrar a pose do robô quando um Aruco é identificado.

Descrevemos acima os procedimentos para encontrar as medições z do FKE utilizando a câmera RGB-D (mais ORB-SLAM 2), a câmera *Tracking* e as câmeras RGB's (para identificar os Arucos). A identificação dos Arucos também poderia ser feita com as câmeras RGB-D e *Tracking*, porém elas estão apontadas para frente e em alguns testes realizados em corredores, elas não conseguiram detectar com robustez os Arucos colados na parede. Portanto, foram incluídas as câmeras RGB's nas laterais do robô (apontadas para parede).

Na Figura 3.12, temos uma ilustração do funcionamento do FKE. O FKE inicia com a crença (μ_0, Σ_0) , e utilizando as ações de controle realiza a etapa de predição. Caso não tenha nenhuma medição, o algoritmo retornará (μ_1, Σ_1) e realizará novamente a predição de forma iterativa. Se no instante de tempo t o robô receber alguma medição do ambiente a etapa de atualização é executada. A incerteza da pose calculada a partir dos Arucos é menor que a incerteza fornecida pela câmeras RGB-D e *Tracking*. Isto se deve ao fato da medição das câmeras RGB-D (mais ORB-SLAM 2) e *Tracking* integrarem erros de acordo com a evolução temporal (erros pequenos, que não se comparam aos da odometria). Portanto, nos momentos que um Aruco for detectado, a sua medição terá precedência e será usada na etapa de atualização. Se as duas câmeras RGB's detectarem Arucos, efetuamos a média entre as poses retornadas por elas. Para aumentar a confiança do sistema de localização, a informação da pose das câmeras RGB-D (mais ORB-SLAM 2) e *Tracking* é “resetada” toda vez que um Aruco for detectado, ou seja, assumimos que a última medição da pose é igual a μ_t . Obs: o FKE não usa as câmeras RGB-D e *Tracking* simultaneamente, a escolha da câmera deve ser feita antes da estimação iniciar.

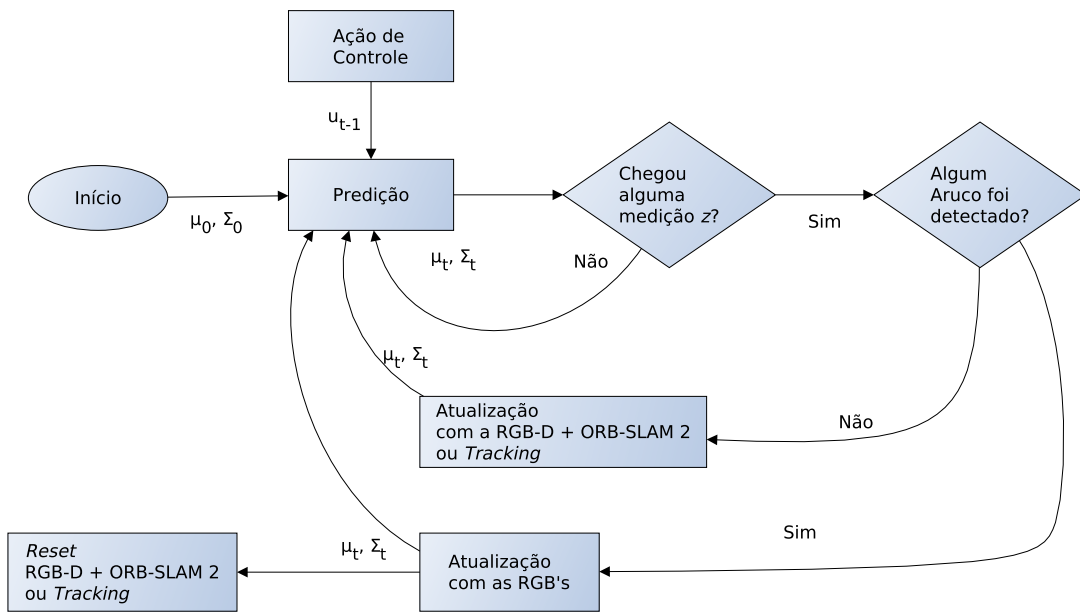


Figura 3.12 – Esquemático do funcionamento do Filtro de Kalman Estendido.

O resultado da estimação da localização da plataforma robótica usando o FKE, em corredores, é apresentado nas Figuras 3.13 e 3.14. Nestas figuras, o robô é lançado na pose inicial $(1, 0, 0)^T$ e se direciona à pose $(33, -27, -\pi/2)^T$. Durante o percurso o robô observa 14 Arucos diferentes, e sabe a posição deles no ambiente. O corredor é um local com poucas *features* (um banco, portas e *banners* na parede), o que se torna uma complicação na estimação da localização usando sensores visuais. Na Figura 3.13, podemos ver que a estimação usando a câmera RGB-D + ORB-SLAM 2 não foi satisfatória, pois houve um fator de escala de aproximadamente 1.8 entre a localização estimada e a real. Indicamos com uma seta pontilhada a última estimação da localização via RGB-D + ORB-SLAM 2 antes de um Aruco ser encontrado. Já o resultado da Figura 3.14 (usando a *Tracking*) foi satisfatório, apresentando um fator de escala de 1.15.

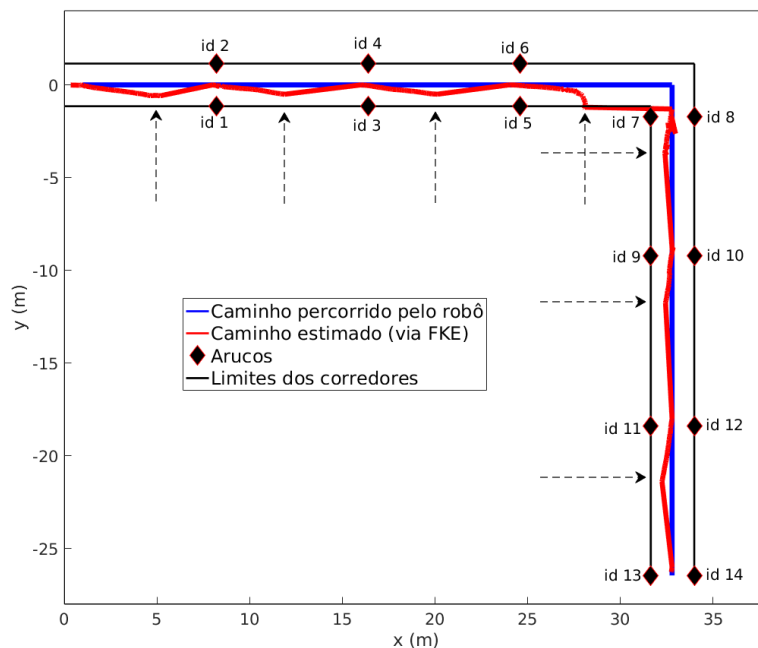


Figura 3.13 – Localização da plataforma robótica em corredores usando câmera RGB-D + ORB-SLAM 2.

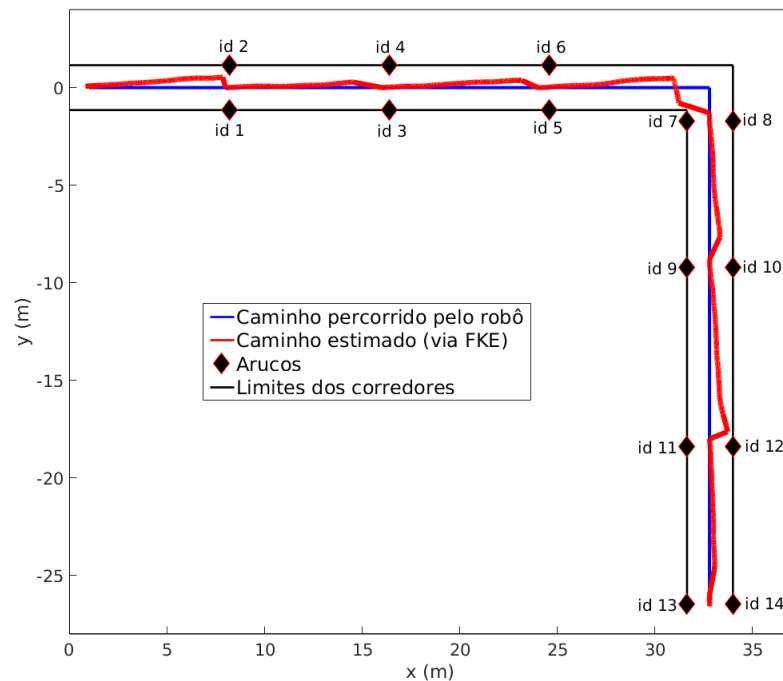


Figura 3.14 – Localização da plataforma robótica em corredores usando a câmera *Tracking*.

3.3.3 Modelo Oculto de Markov

3.3.3.1 Simulação com um robô pontual

Nesta seção, utilizaremos o MOM e o ES-MOM para estimar a localização de um robô pontual diferencial em um ambiente definido topologicamente. O robô possui uma câmera virtual acoplada que tem um ângulo de visão de 360° e alcance limitado. A localização do robô no mapa é inferida a partir da detecção de *landmarks* coloridas presentes no ambiente. O mapa topológico foi construído no Matlab, e é exibido na Figura 3.15. No mapa da Figura 3.15 existem 42 estados simbolizados por “quadrados” e 62 *landmarks* coloridas. As *landmarks* são representadas por “diamantes” de 8 cores (amarela, vermelha, azul, verde, laranja, rosa, marrom e preta). Em um sistema real, os “diamantes” poderiam ser reproduzidos por cones coloridos ou por marcações distinguíveis no ambiente. Sempre que o robô chegar em um estado, ele irá realizar buscas por *landmarks*, e utilizará a câmera virtual para identificar as cores das *landmarks* próximas dele. As linhas azuis mostram as conexões factíveis entre os estados e contêm as informações de distância entre eles. Também há no mapa regiões proibidas (obstáculos) indicadas por retângulos pretos. O mapa da Figura 3.15 pode representar o andar de um prédio, museu, escola e etc.

O mapa topológico é construído *a priori* e armazenado em um banco de dados. Os dados armazenados são os estados, as conexões factíveis entre os estados e o modelo $\lambda = \{\Gamma, B, \zeta\}$. Ao todo, há 13 símbolos de observações possíveis (l_1 a l_{13}), conforme mostra a Tabela 3.1. Por exemplo, o símbolo l_1 é constituído de um “diamante” vermelho e dois amarelos, e pode ser visto nos estados s_1, s_3, s_7, s_9 . Uma particularidade do sistema é a sua simetria e indistinguibilidade. Isto se torna uma dificuldade, pois há muita ambiguidade ao se relacionar uma observação a um estado. Como no exemplo mostrado acima: quando o robô detecta um “diamante” vermelho e dois amarelos ele não consegue distinguir se está no estado s_1, s_3, s_7 , ou s_9 .

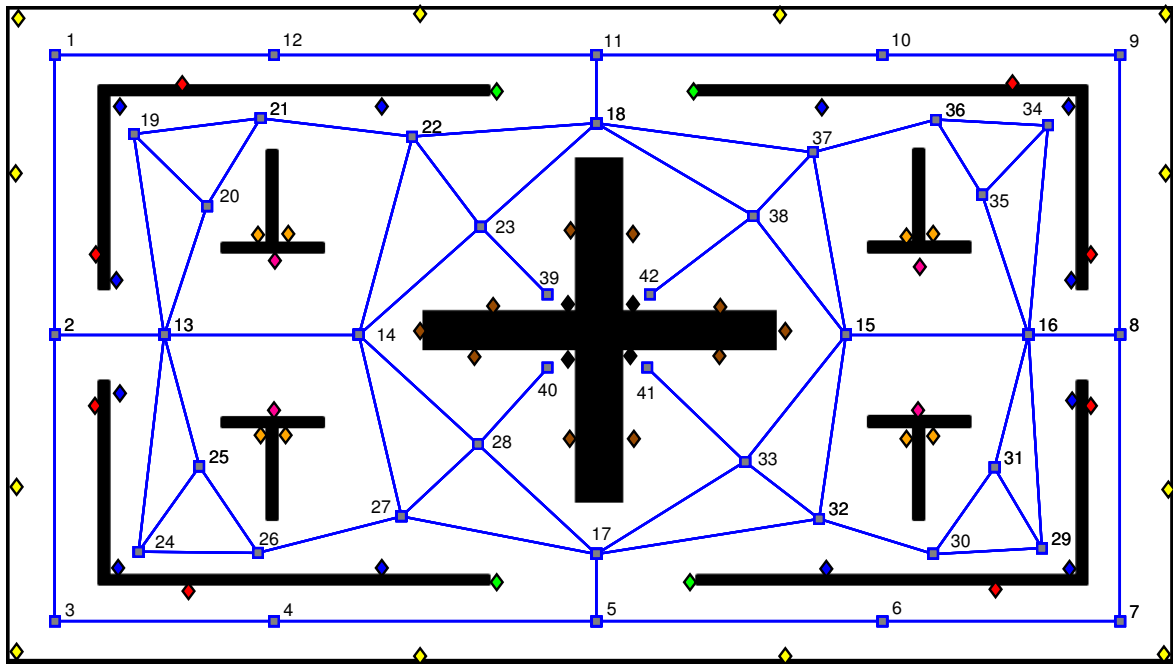


Figura 3.15 – Mapa do ambiente topológico utilizado para simulação do MOM.

Tabela 3.1 – Símbolos de observações referentes a cada estado (simulação).

Símbolos de observações [estados]	Cores das <i>landmakrs</i>							
	Amarela	Azul	Laranja	Marrom	Preta	Rosa	Verde	Vermelha
l_1 [$s_1 - s_3 - s_7 - s_9$]	2	0	0	0	0	0	0	1
l_2 [$s_2 - s_8$]	2	0	0	0	0	0	0	2
l_3 [$s_5 - s_{11}$]	2	0	0	0	0	0	2	0
l_4 [$s_{13} - s_{16}$]	0	2	0	0	0	2	0	0
l_5 [$s_4 - s_6 - s_{10} - s_{12}$]	1	0	0	0	0	0	0	1
l_6 [$s_{17} - s_{18}$]	0	0	0	0	0	0	2	0
l_7 [$s_{14} - s_{15}$]	0	0	0	1	0	2	0	0
l_8 [$s_{19} - s_{24} - s_{29} - s_{34}$]	0	1	0	0	0	0	0	0
l_9 [$s_{20} - s_{25} - s_{31} - s_{35}$]	0	2	1	0	0	0	0	0
l_{10} [$s_{21} - s_{26} - s_{30} - s_{36}$]	0	2	0	0	0	0	0	0
l_{11} [$s_{22} - s_{27} - s_{32} - s_{37}$]	0	1	1	0	0	0	0	0
l_{12} [$s_{23} - s_{28} - s_{33} - s_{38}$]	0	0	0	2	0	0	0	0
l_{13} [$s_{39} - s_{40} - s_{41} - s_{42}$]	0	0	0	2	1	0	0	0

Se o símbolo l_i puder ser visto no estado s_j , teremos $b_j(l_i) = 0.85$ e $b_j(l_k) = 0.15/(M - 1)$ para todos os outros $M - 1$ símbolos de observações l_k possíveis. Este processo é repetido

para todos os estados ($N = 42$) do sistema. Desta forma, utilizando a Tabela 3.1 a matriz de observabilidade B é construída:

$$B = \begin{pmatrix} b_1(l_1) & b_1(l_2) & \cdots & b_1(l_{13}) \\ b_2(l_1) & b_2(l_2) & \cdots & b_2(l_{13}) \\ \vdots & \vdots & \ddots & \vdots \\ b_{42}(l_1) & b_{42}(l_2) & \cdots & b_{42}(l_{13}) \end{pmatrix}. \quad (3.58)$$

3.3.3.1.1 MOM

No MOM, a matriz Γ corresponde à probabilidade de transição de um estado para o outro (sem considerar as ações executadas nos estados). As transições entre estados que não estão conectados na Figura 3.15 recebem peso 0. Já as transições entre estados que estão conectados recebem peso 1. As probabilidades das transições são normalizadas a fim de que $\sum_j \tau_{ij} = 1 \forall i$. Como primeiro teste, supomos que a distribuição inicial de probabilidade ζ é bem definida e que o robô saiu do estado s_2 em direção ao estado s_9 , percorrendo a sequência de estados $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$. Assim, definimos $\zeta_2 = 0.9$ e $\zeta_i = 0.1/(42 - 1)$ para os outros estados s_i do sistema. Também estipulamos $\beta_T(9) = 0.9$ e $\beta_T(i) = 0.1/(42 - 1)$ para os outros estados s_i . As estimativas das variáveis α , β e γ (com base nas observações das *landmarks*), para o referido trajeto, são mostradas nas Tabelas 3.2, 3.3 e 3.4, respectivamente. Nestas tabelas, são exibidos os cinco maiores valores de probabilidade em cada instante de tempo.

Tabela 3.2 – Probabilidade α para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(2) = 0.99$	$\alpha(13) = 0.99$	$\alpha(14) = 0.99$	$\alpha(23) = 0.49$	$\alpha(18) = 0.50$	$\alpha(11) = 0.50$	$\alpha(10) = 0.25$	$\alpha(9) = 0.25$
$\alpha(8) = 0.01$	$\alpha(16) = 0.01$	$\alpha(15) = 0.01$	$\alpha(28) = 0.49$	$\alpha(17) = 0.50$	$\alpha(5) = 0.50$	$\alpha(12) = 0.25$	$\alpha(7) = 0.25$
$\alpha(6) = 0.00$	$\alpha(42) = 0.00$	$\alpha(42) = 0.00$	$\alpha(22) = 0.00$	$\alpha(42) = 0.00$	$\alpha(42) = 0.00$	$\alpha(4) = 0.25$	$\alpha(3) = 0.25$
$\alpha(12) = 0.00$	$\alpha(41) = 0.00$	$\alpha(41) = 0.00$	$\alpha(27) = 0.00$	$\alpha(41) = 0.00$	$\alpha(41) = 0.00$	$\alpha(6) = 0.25$	$\alpha(1) = 0.25$
$\alpha(4) = 0.00$	$\alpha(40) = 0.00$	$\alpha(40) = 0.00$	$\alpha(13) = 0.00$	$\alpha(40) = 0.00$	$\alpha(40) = 0.00$	$\alpha(42) = 0.00$	$\alpha(42) = 0.00$

Tabela 3.3 – Probabilidade β para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
$\beta(2) = 0.08$	$\beta(13) = 0.09$	$\beta(42) = 0.20$	$\beta(11) = 0.20$	$\beta(10) = 0.30$	$\beta(9) = 0.36$	$\beta(10) = 0.36$	$\beta(9) = 0.90$
$\beta(8) = 0.08$	$\beta(23) = 0.09$	$\beta(39) = 0.20$	$\beta(38) = 0.16$	$\beta(12) = 0.30$	$\beta(10) = 0.36$	$\beta(9) = 0.36$	$\beta(42) = 0.00$
$\beta(19) = 0.08$	$\beta(22) = 0.09$	$\beta(18) = 0.14$	$\beta(37) = 0.16$	$\beta(11) = 0.22$	$\beta(11) = 0.27$	$\beta(8) = 0.27$	$\beta(41) = 0.00$
$\beta(20) = 0.08$	$\beta(28) = 0.09$	$\beta(14) = 0.13$	$\beta(23) = 0.16$	$\beta(18) = 0.15$	$\beta(5) = 0.00$	$\beta(2) = 0.00$	$\beta(40) = 0.00$
$\beta(24) = 0.08$	$\beta(27) = 0.09$	$\beta(15) = 0.08$	$\beta(22) = 0.16$	$\beta(6) = 0.00$	$\beta(12) = 0.00$	$\beta(12) = 0.00$	$\beta(39) = 0.00$

Tabela 3.4 – Probabilidade γ para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T bem definidas).

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
$\gamma(2) = 0.99$	$\gamma(13) = 0.99$	$\gamma(14) = 0.99$	$\gamma(23) = 0.99$	$\gamma(18) = 0.99$	$\gamma(11) = 0.99$	$\gamma(10) = 0.99$	$\gamma(9) = 0.99$
$\gamma(8) = 0.01$	$\gamma(16) = 0.01$	$\gamma(15) = 0.01$	$\gamma(28) = 0.01$	$\gamma(17) = 0.01$	$\gamma(5) = 0.01$	$\gamma(12) = 0.00$	$\gamma(1) = 0.00$
$\gamma(42) = 0.00$	$\gamma(42) = 0.00$	$\gamma(42) = 0.00$	$\gamma(38) = 0.00$	$\gamma(42) = 0.00$	$\gamma(42) = 0.00$	$\gamma(4) = 0.00$	$\gamma(3) = 0.00$
$\gamma(41) = 0.00$	$\gamma(41) = 0.00$	$\gamma(41) = 0.00$	$\gamma(33) = 0.00$	$\gamma(41) = 0.00$	$\gamma(41) = 0.00$	$\gamma(6) = 0.00$	$\gamma(7) = 0.00$
$\gamma(40) = 0.00$	$\gamma(40) = 0.00$	$\gamma(40) = 0.00$	$\gamma(42) = 0.00$	$\gamma(40) = 0.00$	$\gamma(40) = 0.00$	$\gamma(42) = 0.00$	$\gamma(42) = 0.00$

Com uma boa confiança inicial, a variável α apresentou bons resultados nas três primeiras iterações. Já na quarta iteração, a observação coletada pode ser vista nos estados s_{23} e s_{28} , causando uma ambiguidade ao estimador. Nas demais iterações o procedimento *forward* também não conseguiu distinguir a ambiguidade presente no mapa topológico. Os valores da variável β não foram muito precisos, e em várias iterações aconteceram erros. Apesar das variáveis α e β não apresentarem bons resultados de forma isolada, a combinação delas γ é robusta para estimação da localização, como pode ser verificado na Tabela 3.4.

Um segundo teste mais conservador foi realizado usando o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$. Todavia, considerando que o robô não tem nenhuma informação prévia sobre o trajeto a ser percorrido. Assim, teremos distribuições uniformes ζ_i e $\beta_T(i)$ para todos os estados s_i do sistema. As estimativas das variáveis α , β e γ , para o trajeto sem informação prévia, são mostradas nas Tabelas 3.5, 3.6 e 3.7, respectivamente.

Tabela 3.5 – Probabilidade α para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(2)=0.43$	$\alpha(13)=0.50$	$\alpha(14)=0.50$	$\alpha(23)=0.25$	$\alpha(18)=0.50$	$\alpha(11)=0.50$	$\alpha(10)=0.25$	$\alpha(9)=0.25$
$\alpha(8)=0.43$	$\alpha(16)=0.50$	$\alpha(15)=0.50$	$\alpha(28)=0.25$	$\alpha(17)=0.50$	$\alpha(5)=0.50$	$\alpha(12)=0.25$	$\alpha(7)=0.25$
$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(33)=0.25$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(4)=0.25$	$\alpha(3)=0.25$
$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(38)=0.25$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(6)=0.25$	$\alpha(1)=0.25$
$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$

Tabela 3.6 – Probabilidade β para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
$\beta(2)=0.08$	$\beta(13)=0.09$	$\beta(42)=0.10$	$\beta(11)=0.10$	$\beta(10)=0.15$	$\beta(11)=0.14$	$\beta(2)=0.13$	$\beta(42)=0.02$
$\beta(8)=0.08$	$\beta(23)=0.09$	$\beta(41)=0.10$	$\beta(38)=0.10$	$\beta(12)=0.15$	$\beta(5)=0.14$	$\beta(8)=0.13$	$\beta(42)=0.02$
$\beta(19)=0.08$	$\beta(22)=0.09$	$\beta(40)=0.10$	$\beta(37)=0.08$	$\beta(6)=0.15$	$\beta(12)=0.09$	$\beta(12)=0.09$	$\beta(40)=0.02$
$\beta(20)=0.08$	$\beta(28)=0.09$	$\beta(39)=0.10$	$\beta(23)=0.08$	$\beta(4)=0.15$	$\beta(10)=0.09$	$\beta(10)=0.09$	$\beta(39)=0.02$
$\beta(24)=0.08$	$\beta(27)=0.09$	$\beta(18)=0.06$	$\beta(22)=0.08$	$\beta(11)=0.11$	$\beta(9)=0.09$	$\beta(9)=0.09$	$\beta(38)=0.02$

Tabela 3.7 – Probabilidade γ para o trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$ (com ζ e β_T desconhecidas).

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
$\gamma(2)=0.50$	$\gamma(13)=0.50$	$\gamma(14)=0.50$	$\gamma(23)=0.25$	$\gamma(18)=0.50$	$\gamma(11)=0.50$	$\gamma(10)=0.25$	$\gamma(9)=0.25$
$\gamma(8)=0.50$	$\gamma(16)=0.50$	$\gamma(15)=0.50$	$\gamma(28)=0.25$	$\gamma(17)=0.50$	$\gamma(5)=0.50$	$\gamma(12)=0.25$	$\gamma(1)=0.25$
$\gamma(42)=0.00$	$\gamma(42)=0.00$	$\gamma(42)=0.00$	$\gamma(38)=0.25$	$\gamma(42)=0.00$	$\gamma(42)=0.00$	$\gamma(4)=0.25$	$\gamma(3)=0.25$
$\gamma(41)=0.00$	$\gamma(41)=0.00$	$\gamma(41)=0.00$	$\gamma(33)=0.25$	$\gamma(41)=0.00$	$\gamma(41)=0.00$	$\gamma(6)=0.25$	$\gamma(7)=0.25$
$\gamma(40)=0.00$	$\gamma(40)=0.00$	$\gamma(40)=0.00$	$\gamma(42)=0.00$	$\gamma(40)=0.00$	$\gamma(40)=0.00$	$\gamma(42)=0.00$	$\gamma(42)=0.00$

A falta de conhecimento prévio sobre a configuração inicial e final do robô refletiu nas variáveis α e β . As probabilidades α 's da Tabela 3.5 indicam os estados onde as observações o_t coletadas pelo robô podem ser encontradas, mas não têm confiança suficiente para indicar o real estado do robô. Como os valores de β na Tabela 3.6 foram incertos, consequentemente a estimativa de γ também foi incerta, vide Tabela 3.7. O algoritmo 3 (Viterbi) foi implementado para os dois testes, e em ambos não conseguiu identificar o real percurso executado pelo robô, a partir do modelo λ e da sequência de observações. No primeiro teste, a sequência mais provável

apontada pelo algoritmo de Viterbi foi $q^* = [s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{12} - s_1]$, e no segundo $q^* = [s_2 - s_{13} - s_{14} - s_{28} - s_{17} - s_5 - s_6 - s_7]$.

De acordo com os dois testes acima, podemos ver que o conhecimento da localização inicial e final é relevante para o rastreamento dos estados do robô. É importante salientar que as variáveis β e γ poderão ser utilizadas somente no pós-processamento da localização, ou seja, quando todas as observações tiverem sido coletadas. Para estimar a localização em tempo real, somente a variável α poderá ser usada, e como visto nos dois testes, ela sozinha não foi suficiente. Na sequência, vamos estimar a localização do mesmo trajeto $[s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9]$, porém usando o ES-MOM e incorporando na probabilidade de transição de estados as ações de controle (contínuas) executadas entre os estados.

3.3.3.1.2 ES-MOM

Para que o robô execute o movimento de um estado para outro é necessário um planejador de movimento local. Neste trabalho, desenvolvemos um planejador local que retorna as velocidades $u = (v, \omega)^T$ que devem ser aplicadas ao robô diferencial para que ele se desloque entre os estados, detectando as *landmarks* do ambiente. O planejador faz uso do controlador *feedback linearization* e pode ser visto com mais detalhes no Apêndice A. Cada ação discreta a , que leva o robô de um estado para outro, engloba uma série de ações contínuas $\mathbb{U} = (u_1, \dots, u_{|\mathbb{U}|})$.

No MOM, a probabilidade de transição de estados era definida por $\tau_{ij} = p(q_t = s_j | q_{t-1} = s_i)$ e era armazenada na matriz Γ . O planejador local também foi utilizado no MOM, mas as ações executadas não foram usadas nas estimativas de localização. Já no ES-MOM, a probabilidade de transição é dada por $p(q_t = s_j | q_{t-1} = s_i, a_{t-1}) = p(q_t = s_j | q_{t-1} = s_i, \mathbb{U}_{t-1})$. Executando a sequência de ações \mathbb{U}_{t-1} no estado s_i , e utilizando o modelo de propagação de movimento (contínuo) apresentado na Equação (3.52), podemos prever em qual local do mapa o robô estará⁷. Este local é simbolizado por \bar{s} e será um estado “virtual” no mapa topológico, ou seja, um artifício para estimar a probabilidade de transição, como mostra a Figura 3.16.

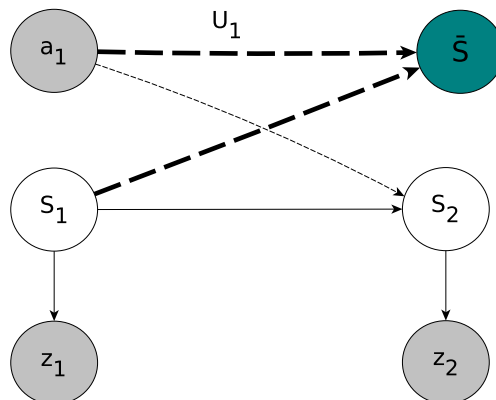


Figura 3.16 – Modelo gráfico do ES-MOM incorporando o estado virtual \bar{s} , utilizado para calcular a probabilidade de transição de estados $p(q_t = s_j | q_{t-1} = s_i, \mathbb{U}_{t-1})$.

⁷ O estado s_i é uma posição $(x, y)^T$, portanto, vamos adicionar uma orientação “fictícia” ao estado s_i para aplicar a Equação (3.52). Esta orientação será a que o robô diferencial possuía antes de executar a sequência \mathbb{U}_{t-1} .

Desta feita, a probabilidade de transição de estados pode ser determinada por [Llarena et al., 2012]:

$$p(q_t = s_j | q_{t-1} = s_i, \mathbb{U}_{t-1}) = \frac{D(\bar{s}, s_j)}{\sum_{j=1}^N D(\bar{s}, s_j)}, \quad (3.59)$$

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

$$D(\bar{s}, s_j) = e^{-\frac{1}{2} \left(\frac{(s_{jx} - \bar{s}_x)^2}{\sigma^2} + \frac{(s_{jy} - \bar{s}_y)^2}{\sigma^2} \right)}. \quad (3.60)$$

O somatório no denominador da Equação (3.59) é necessário para normalizar a probabilidade de transição. O estado s_j é uma posição $(x, y)^T$ no mapa, por isto usamos somente as componentes x e y de \bar{s} . Como a probabilidade de transição é dada em função de \mathbb{U} , ela deverá ser calculada em cada iteração, não podendo ser calculada *a priori* e armazenada em Γ . Utilizando a probabilidade de transição da Equação (3.59) e assumindo que o robô não tem conhecimento prévio sobre a localização inicial⁸, o procedimento *forward* do ES-MOM foi implementado para estimar a localização do robô no mapa da Figura 3.15. O trajeto percorrido foi o mesmo dos testes anteriores [$s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9$]. Os resultados para três valores de σ são mostrados nas Tabelas 3.8, 3.9 e 3.10.

Tabela 3.8 – Probabilidade α do ES-MOM para o trajeto [$s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9$] (com σ).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(2)=0.43$	$\alpha(13)=0.78$	$\alpha(14)=0.99$	$\alpha(23)=0.85$	$\alpha(18)=0.94$	$\alpha(11)=0.95$	$\alpha(10)=0.99$	$\alpha(9)=0.99$
$\alpha(8)=0.43$	$\alpha(16)=0.21$	$\alpha(15)=0.01$	$\alpha(28)=0.11$	$\alpha(17)=0.06$	$\alpha(5)=0.05$	$\alpha(12)=0.01$	$\alpha(1)=0.01$
$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(33)=0.02$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(4)=0.00$	$\alpha(3)=0.00$
$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(38)=0.02$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(6)=0.00$	$\alpha(7)=0.00$
$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$

Tabela 3.9 – Probabilidade α do ES-MOM para o trajeto [$s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9$] (com 0.1σ).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(2)=0.43$	$\alpha(13)=0.83$	$\alpha(14)=0.99$	$\alpha(23)=0.91$	$\alpha(18)=0.97$	$\alpha(11)=0.99$	$\alpha(10)=0.99$	$\alpha(9)=0.99$
$\alpha(8)=0.43$	$\alpha(16)=0.16$	$\alpha(15)=0.01$	$\alpha(28)=0.06$	$\alpha(17)=0.03$	$\alpha(5)=0.01$	$\alpha(12)=0.01$	$\alpha(1)=0.01$
$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(33)=0.01$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(4)=0.00$	$\alpha(3)=0.00$
$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(38)=0.01$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(6)=0.00$	$\alpha(7)=0.00$
$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$

Tabela 3.10 – Probabilidade α do ES-MOM para o trajeto [$s_2 - s_{13} - s_{14} - s_{23} - s_{18} - s_{11} - s_{10} - s_9$] (com 0.01σ).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(2)=0.43$	$\alpha(13)=0.95$	$\alpha(14)=0.99$	$\alpha(23)=0.95$	$\alpha(18)=0.98$	$\alpha(11)=0.99$	$\alpha(10)=0.99$	$\alpha(9)=0.99$
$\alpha(8)=0.43$	$\alpha(16)=0.05$	$\alpha(15)=0.01$	$\alpha(28)=0.05$	$\alpha(17)=0.02$	$\alpha(5)=0.01$	$\alpha(12)=0.01$	$\alpha(1)=0.01$
$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(33)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(4)=0.00$	$\alpha(3)=0.00$
$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(38)=0.00$	$\alpha(41)=0.00$	$\alpha(41)=0.00$	$\alpha(6)=0.00$	$\alpha(7)=0.00$
$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$

⁸ O único conhecimento é que o robô começa a simulação em um estado arbitrário, e que inicialmente está orientado de acordo com o sistema de referência inercial do mapa.

Pelas Tabelas 3.8, 3.9 e 3.10, podemos ver que as estimativas melhoram consideravelmente com a inclusão da sequência de controle \mathbb{U} . O algoritmo *forward* do ES-MOM conseguiu convergir rapidamente para localização real mesmo sem usar as variáveis β e γ . Logo, podemos concluir que o ES-MOM é mais robusto que o MOM para lidar com a estimativa da localização em sistemas ambíguos e simétricos, e pode ser utilizado em tempo real. Analisando as tabelas, a influência do parâmetro σ pode ser percebida. Conforme os valores de σ são decrescidos, a variância da PDF gaussiana da Equação (3.60) fica mais próxima da média \bar{s} , fazendo com que estados mais próximos de \bar{s} recebam maior probabilidade, refinando a estimativa do ES-MOM.

O ES-MOM também foi testado para o percurso $[s_{30} - s_{32} - s_{17} - s_{27} - s_{14} - s_{22} - s_{23} - s_{39}]$, e o resultado é exibido na Tabela 3.11. Apesar deste trajeto ser muito ambíguo (maioria das *landmarks* coletadas podem ser vistas em quarto lugares do mapa), o ES-MOM conseguiu contornar as incertezas presentes no cenário.

Tabela 3.11 – Probabilidade α do ES-MOM para o trajeto $[s_{30} - s_{32} - s_{17} - s_{27} - s_{14} - s_{22} - s_{23} - s_{39}]$ (com σ).

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$\alpha(30)=0.22$	$\alpha(32)=0.69$	$\alpha(17)=0.88$	$\alpha(27)=0.93$	$\alpha(14)=0.98$	$\alpha(22)=0.99$	$\alpha(23)=1.00$	$\alpha(39)=1.00$
$\alpha(36)=0.22$	$\alpha(37)=0.25$	$\alpha(18)=0.11$	$\alpha(22)=0.07$	$\alpha(15)=0.02$	$\alpha(27)=0.01$	$\alpha(28)=0.00$	$\alpha(40)=0.00$
$\alpha(26)=0.22$	$\alpha(27)=0.03$	$\alpha(42)=0.00$	$\alpha(32)=0.00$	$\alpha(42)=0.00$	$\alpha(37)=0.00$	$\alpha(38)=0.00$	$\alpha(42)=0.00$
$\alpha(21)=0.22$	$\alpha(22)=0.02$	$\alpha(41)=0.00$	$\alpha(37)=0.00$	$\alpha(41)=0.00$	$\alpha(32)=0.00$	$\alpha(33)=0.00$	$\alpha(41)=0.00$
$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(40)=0.00$	$\alpha(42)=0.00$	$\alpha(42)=0.00$	$\alpha(38)=0.00$

3.3.3.2 Experimento com robô real (ES-MOM)

Nas simulações apresentadas na Seção 3.3.3.1, as *landmarks* utilizadas eram marcações coloridas no ambiente. Nesta seção, mostraremos experimentos realizados com o ES-MOM no mapa topológico da Figura 3.17, no qual Arucos são usados como *landmarks*. Este mapa é uma abstração topológica do 2º andar do Bloco 1 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 3.12. As linhas azuis mostram as conexões factíveis entre os estados. Podemos observar que este ambiente é simétrico e há ambiguidade ao relacionar uma observação a um estado, portanto, construímos a matriz de observabilidade B de forma análoga a da Seção 3.3.3.1.

Tabela 3.12 – Símbolos de observações (id's) referentes a cada estado (experimento).

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

Na implementação da simulação do ES-MOM na Seção 3.3.3.1.2, utilizamos apenas o modelo de propagação da Equação (3.52) para estimar o estado \bar{s} , ou seja, a etapa de predição do FKE. O resultado apresentado foi bom, dado que era uma simulação e as incertezas eram

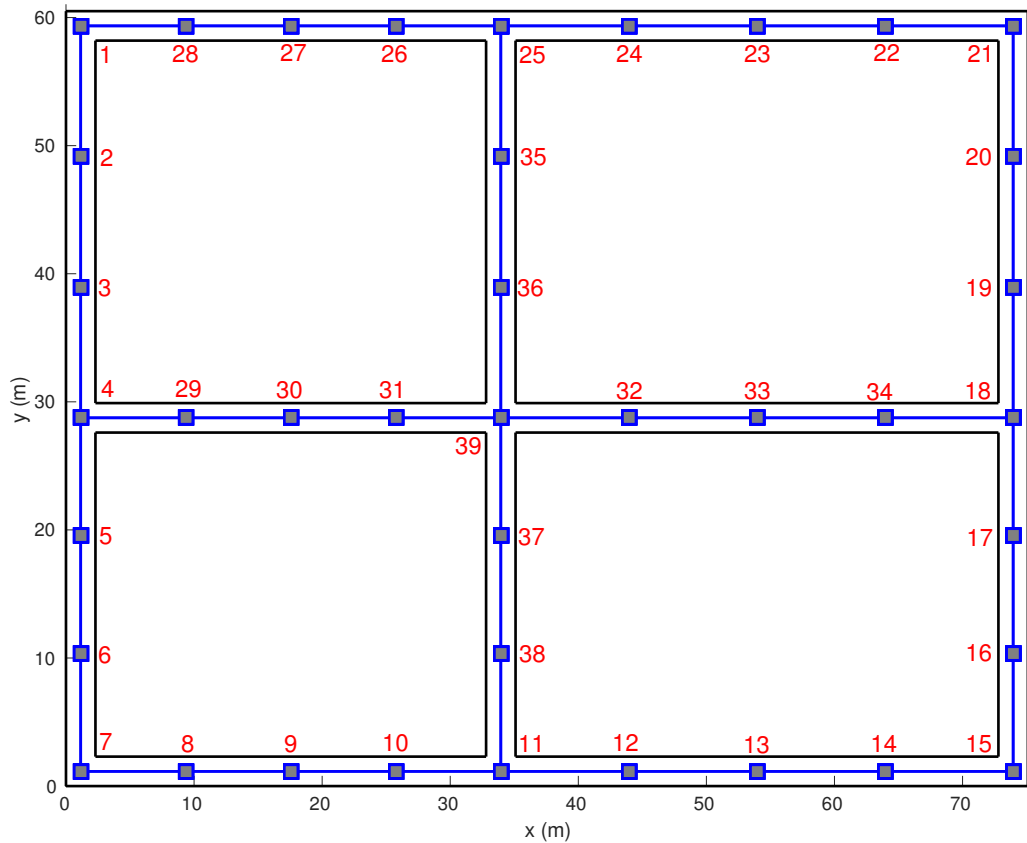


Figura 3.17 – Mapa topológico (em escala real) utilizado nos experimentos do ES-MOM.

controladas. Entretanto, em um ambiente real, estimar o estado \bar{s} somente com as ações \mathbb{U} é impreciso, assim, estimaremos o estado \bar{s} usando o FKE completo (com previsão e atualização). Na fase de atualização, as medições usadas são oriundas da câmera *Tracking* (o ambiente é constituído de corredores com poucas *features*, neste cenário a câmera *Tracking* é mais robusta que a RGB-D). Como a id do Aruco identificado pode estar em mais de um lugar do mapa, ele só será usado no FKE quando o ES-MOM tiver uma certeza maior que 70% sobre o estado atual. Assim, quando a certeza for menor que este percentual, o Aruco será usado somente para detectar que o sistema chegou em um novo estado.

Desta feita, aplicamos o procedimento *forward* do ES-MOM para estimar a localização do robô (Figura 3.8) no trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$, assumindo que o robô começa o experimento em um estado (mas ele não sabe qual) e inicialmente está orientado de acordo com o sistema de referência inercial do mapa. O resultado é mostrado na Tabela 3.13. Foram realizados alguns testes com o planejador de movimento apresentado no Apêndice A, porém o robô (Figura 3.8) é muito baixo e não consegue percorrer todos os lugares do ambiente sozinho. Existem pequenas elevações no piso que o robô não consegue ultrapassar de forma autônoma, sendo necessária a intervenção humana. Portanto, o robô foi guiado neste percurso através de um *joystick*. Um vídeo ilustrativo deste experimento pode ser visualizado em https://youtu.be/qE_47GGwGA.

Tabela 3.13 – Probabilidade α do ES-MOM para o trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$.

α_1	$\alpha(1)=0.21$	$\alpha(7)=0.21$	$\alpha(15)=0.21$	$\alpha(21)=0.21$	$\alpha(38)=0.00$	$\alpha(39)=0.00$
α_2	$\alpha(8)=0.30$	$\alpha(28)=0.30$	$\alpha(29)=0.30$	$\alpha(14)=0.02$	$\alpha(22)=0.02$	$\alpha(34)=0.02$
α_3	$\alpha(9)=0.32$	$\alpha(27)=0.32$	$\alpha(30)=0.32$	$\alpha(13)=0.01$	$\alpha(23)=0.01$	$\alpha(33)=0.01$
α_4	$\alpha(10)=0.33$	$\alpha(26)=0.33$	$\alpha(31)=0.33$	$\alpha(12)=0.00$	$\alpha(24)=0.00$	$\alpha(32)=0.00$
α_5	$\alpha(11)=0.48$	$\alpha(25)=0.48$	$\alpha(31)=0.00$	$\alpha(10)=0.00$	$\alpha(26)=0.00$	$\alpha(37)=0.00$
α_6	$\alpha(38)=0.94$	$\alpha(35)=0.06$	$\alpha(11)=0.00$	$\alpha(10)=0.00$	$\alpha(26)=0.00$	$\alpha(24)=0.00$
α_7	$\alpha(37)=0.98$	$\alpha(36)=0.02$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(35)=0.00$	$\alpha(34)=0.00$
α_8	$\alpha(39)=0.99$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$	$\alpha(35)=0.00$	$\alpha(34)=0.00$
α_9	$\alpha(31)=0.99$	$\alpha(32)=0.00$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$
α_{10}	$\alpha(30)=0.99$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$	$\alpha(35)=0.00$
α_{11}	$\alpha(29)=0.99$	$\alpha(31)=0.00$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$
α_{12}	$\alpha(4)=0.99$	$\alpha(18)=0.00$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$
α_{13}	$\alpha(5)=0.95$	$\alpha(3)=0.05$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$	$\alpha(36)=0.00$
α_{14}	$\alpha(6)=0.99$	$\alpha(2)=0.00$	$\alpha(4)=0.00$	$\alpha(39)=0.00$	$\alpha(38)=0.00$	$\alpha(37)=0.00$

Apesar do mapa da Figura 3.17 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 (mesmo sem usar as variáveis β e γ). Para efeito de comparação, mostramos o resultado do MOM para o mesmo trajeto na Tabela 3.14, na qual é notável o aumento da quantidade de instantes de conflitos de localização (conforme as simulações da Seção 3.3.3.1.1). Por outro lado, o ES-MOM conseguiu contornar as incertezas do sistema e mostra ser um método de estimação de localização robusto.

Tabela 3.14 – Probabilidade α do MOM para o trajeto $[s_7 - s_8 - s_9 - s_{10} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_5 - s_6]$.

α_1	$\alpha(1)=0.21$	$\alpha(7)=0.21$	$\alpha(15)=0.21$	$\alpha(21)=0.21$	$\alpha(39)=0.00$	$\alpha(38)=0.00$
α_2	$\alpha(8)=0.15$	$\alpha(14)=0.15$	$\alpha(22)=0.15$	$\alpha(28)=0.15$	$\alpha(29)=0.11$	$\alpha(34)=0.11$
α_3	$\alpha(9)=0.16$	$\alpha(13)=0.16$	$\alpha(23)=0.16$	$\alpha(27)=0.16$	$\alpha(30)=0.15$	$\alpha(33)=0.15$
α_4	$\alpha(10)=0.16$	$\alpha(12)=0.16$	$\alpha(24)=0.16$	$\alpha(26)=0.16$	$\alpha(31)=0.15$	$\alpha(32)=0.15$
α_5	$\alpha(11)=0.45$	$\alpha(25)=0.45$	$\alpha(39)=0.01$	$\alpha(9)=0.01$	$\alpha(23)=0.00$	$\alpha(27)=0.00$
α_6	$\alpha(35)=0.46$	$\alpha(38)=0.46$	$\alpha(10)=0.01$	$\alpha(12)=0.01$	$\alpha(24)=0.01$	$\alpha(26)=0.01$
α_7	$\alpha(36)=0.47$	$\alpha(37)=0.47$	$\alpha(11)=0.01$	$\alpha(25)=0.01$	$\alpha(35)=0.01$	$\alpha(38)=0.01$
α_8	$\alpha(39)=0.94$	$\alpha(35)=0.01$	$\alpha(36)=0.01$	$\alpha(37)=0.01$	$\alpha(38)=0.01$	$\alpha(25)=0.00$
α_9	$\alpha(31)=0.48$	$\alpha(32)=0.48$	$\alpha(36)=0.01$	$\alpha(37)=0.01$	$\alpha(39)=0.01$	$\alpha(26)=0.00$
α_{10}	$\alpha(30)=0.47$	$\alpha(33)=0.47$	$\alpha(39)=0.02$	$\alpha(31)=0.01$	$\alpha(32)=0.01$	$\alpha(27)=0.00$
α_{11}	$\alpha(29)=0.47$	$\alpha(34)=0.47$	$\alpha(31)=0.01$	$\alpha(32)=0.01$	$\alpha(30)=0.01$	$\alpha(33)=0.01$
α_{12}	$\alpha(4)=0.47$	$\alpha(18)=0.47$	$\alpha(29)=0.01$	$\alpha(30)=0.01$	$\alpha(33)=0.01$	$\alpha(34)=0.01$
α_{13}	$\alpha(3)=0.24$	$\alpha(5)=0.24$	$\alpha(17)=0.24$	$\alpha(19)=0.24$	$\alpha(29)=0.01$	$\alpha(34)=0.01$
α_{14}	$\alpha(2)=0.24$	$\alpha(6)=0.24$	$\alpha(16)=0.24$	$\alpha(20)=0.24$	$\alpha(4)=0.01$	$\alpha(18)=0.01$

Realizamos o teste da oclusão com o ES-MOM no trajeto $[s_{15} - s_{14} - s_{13} - s_{12} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_3 - s_2 - s_1]$. A oclusão de uma *landmark* em um problema real pode ocorrer quando um objeto é deixado em frente a ela, ou quando (por algum motivo) o robô não consegue identificá-la no momento em que passa por um estado. Neste experimento, inserimos o erro de oclusão nos estados s_4 e s_{37} , ou seja, o Aruco próximo a cada um destes estados foi tampado e não poderá ser observado pelo robô. O resultado da oclusão das *landmarks* é mostrado na Tabela 3.15, na qual é possível notar que o ES-MOM consegue se recuperar dos erros de oclusão quando se tem uma boa confiança sobre o estado do robô. Um vídeo ilustrativo deste experimento pode ser visto em https://youtu.be/Cj1T_rC3LEs.

Tabela 3.15 – Probabilidade α do ES-MOM para o trajeto $[s_{15} - s_{14} - s_{13} - s_{12} - s_{11} - s_{38} - s_{37} - s_{39} - s_{31} - s_{30} - s_{29} - s_4 - s_3 - s_2 - s_1]$ com oclusões das *landmarks* nos estados s_4 e s_{37} .

α_1	$\alpha(1)= 0.21$	$\alpha(7)= 0.21$	$\alpha(15)= 0.21$	$\alpha(21)= 0.21$	$\alpha(38)= 0.00$	$\alpha(39)= 0.00$
α_2	$\alpha(14)= 0.30$	$\alpha(22)= 0.30$	$\alpha(34)= 0.30$	$\alpha(8)= 0.02$	$\alpha(28)= 0.02$	$\alpha(29)= 0.02$
α_3	$\alpha(13)= 0.32$	$\alpha(23)= 0.32$	$\alpha(33)= 0.32$	$\alpha(9)= 0.01$	$\alpha(27)= 0.01$	$\alpha(30)= 0.01$
α_4	$\alpha(12)= 0.33$	$\alpha(24)= 0.33$	$\alpha(32)= 0.33$	$\alpha(10)= 0.00$	$\alpha(26)= 0.00$	$\alpha(31)= 0.00$
α_5	$\alpha(11)= 0.48$	$\alpha(25)= 0.48$	$\alpha(39)= 0.01$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$
α_6	$\alpha(38)= 0.94$	$\alpha(35)= 0.06$	$\alpha(39)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_7	$\alpha(39)= 0.99$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$	$\alpha(34)= 0.00$
α_8	$\alpha(31)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_9	$\alpha(30)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_{10}	$\alpha(29)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_{11}	$\alpha(3)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_{12}	$\alpha(2)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$
α_{13}	$\alpha(1)= 0.99$	$\alpha(39)= 0.00$	$\alpha(38)= 0.00$	$\alpha(37)= 0.00$	$\alpha(36)= 0.00$	$\alpha(35)= 0.00$

3.4 Conclusões do capítulo

Neste capítulo, apresentamos o Modelo Oculto de Markov (MOM), método capaz de localizar robôs móveis em ambientes topológicos internos. Neste método, a inferência do estado que o robô está localizado é feita de forma indireta, por meio da observação de *landmarks* artificiais adicionadas nos ambientes propostos. O MOM foi utilizado para determinar o estado mais provável do robô, em tempo real, por meio do procedimento *forward*. Também apresentamos o ES-MOM, variação do MOM, que utiliza a estimação da localização no espaço de estados contínuo (via FKE) para determinar a probabilidade de transição de estados. Os resultados obtidos com o ES-MOM foram mais satisfatórios que os do MOM, e a sua robustez pôde ser comprovada pelas simulações e experimentos realizados.

INTEGRAÇÃO ENTRE LOCALIZAÇÃO E PLANEJAMENTO DE MOVIMENTO DE ROBÔS EM AMBIENTES INTERNOS

Os sistemas robóticos são inerentemente carregados de incertezas. Na perspectiva da navegação de robôs móveis, as incertezas podem ser divididas em dois grupos: incerteza na percepção e incerteza nos efeitos das ações de controle. As incertezas da percepção estão relacionadas ao fato dos sensores do robô não conseguirem mensurar completamente os estados do ambiente, e que as medições coletadas são projeções ruidosas do verdadeiro estado do robô. Algoritmos probabilísticos capazes de lidar com a incerteza da percepção na estimação da localização de robôs móveis foram apresentados no Capítulo 3. As ações executadas por um robô também têm incertezas, dado que os resultados das ações de controle são não determinísticos. Deste modo, o robô deve levar em conta as incertezas da percepção do ambiente e das suas ações quando tiver que tomar uma decisão [Thrun et al., 2005].

O problema da navegação de robôs móveis em ambientes incertos pode ser resolvido com o Processo de Decisão de Markov - PDM (do inglês, *Markov Decision Process*) e o com Processo de Decisão de Markov Parcialmente Observável - PDMPO (do inglês, *Partially Observable Markov Decision Process*). Estes métodos probabilísticos selecionam uma sequência de ações com o objetivo de realizar uma dada tarefa, considerando as imprecisões do sistema [Pineau et al., 2004]. Diferente do MOM, no PDM há possibilidade de um agente intervir no sistema executando ações. O PDM modela sistemas nos quais os estados podem ser completamente observados sempre que uma decisão é tomada, mas cada decisão possui um resultado incerto. Para resolver um PDM deve-se encontrar a política ótima que determina qual decisão tomar para maximizar a recompensa esperada [Pellegrini and Wainer, 2007]. O PDMPO é uma extensão do PDM, no qual, os estados não podem ser medidos diretamente, mas são inferidos por meio de observações indiretas [Páll et al., 2016]. Por ser uma estrutura que abrange uma maior quantidade

de problemas reais, e incorporar a incerteza sobre as percepções e ações, o PDMPO será o objeto de estudo principal deste capítulo.

Neste capítulo, nos concentraremos nas aplicações robóticas de planejamento de caminhos para ambientes topológicos incertos. Algumas pesquisas recentes neste contexto são mostradas na sequência. No trabalho de Páll et al. [2016], o PDMPO é utilizado em um problema de assistência doméstica em um ambiente de trabalho discretizado. Nesta aplicação, um robô monitora os estados parcialmente observáveis de interruptores, e os desliga se necessário. Wang et al. [2018] utilizam o PDMPO para realizar buscas de objetos em ambientes que possuem a estrutura topológica. No planejamento do caminho há uma ponderação entre o custo do caminho executado e o ganho de informações no processo de busca. Já Nardi and Stachniss [2019], utilizam uma aproximação do PDMPO para planejamento de caminhos de um robô em uma rede rodoviária, resultando em políticas de navegação relacionadas ao grau de incerteza das ações, percepções e crenças.

Desta forma, apresentamos uma introdução ao PDM na Seção 4.1, e a descrição da integração entre localização e planejamento (PDMPO) na Seção 4.2. Uma simulação envolvendo o controle das ações de um robô móvel terrestre que se direciona a uma posição previamente selecionada de um galpão de estoque, utilizando o PDMPO, é discutida na Seção 4.3. Por fim, mostramos experimentos reais realizados em um ambiente de convivência usando o PDMPO (Seção 4.4), e as conclusões do capítulo na Seção 4.5. Os métodos apresentados neste capítulo consideram que os estados, ações, observações e tempo são discretos.

4.1 Processo de Decisão de Markov (PDM)

Os trabalhos de Bellman [1957b] e Howard [1960] são considerados como ponto de partida dos Processos de Decisão de Markov (PDM). Desde sua introdução na década de 1950, vários resultados sobre o PDM têm sido publicados em periódicos, livros e teses. Os PDM's também são conhecidos como programação dinâmica estocástica, e modelam problemas dinâmicos de tomada de decisão sequencial que têm resultados incertos [Hu and Yue, 2007]. Segundo Feinberg and Shwartz [2012], o PDM é tradicionalmente aplicado em problemas de: roteamento (encontrar o menor caminho até um estado alvo), manutenção e reparo, estoque, controle de filas e agendamento estocástico.

Os PDM's assumem que os estados do ambiente podem ser totalmente observados em todo instante de tempo, e que o modelo de percepção $p(o|s)$ é determinístico. No entanto, as transições entre os estados $p(s'|a, s)$ são probabilísticas e é possível interferir no processo periodicamente (em “épocas de decisão”) executando ações. Quando o tomador de decisões executa uma ação em um estado, uma recompensa é recebida e o sistema evolui possivelmente para outro estado diferente. O PDM é qualificado como markoviano, pois a probabilidade de transição depende apenas do estado e da ação atual do sistema. O PDM também é dito “de

decisão”, pois há possibilidade de um agente intervir no sistema por meio das ações (o que diferencia o PDM das cadeias de Markov). Em suma, no PDM o modelo de percepção $p(o|s)$ é determinístico e o modelo de transição de estados $p(s'|a,s)$ é estocástico [Pellegrini and Wainer, 2007; Puterman, 2014; Thrun et al., 2005].

Um PDM pode ser formalmente definido pela tupla (S, A, Γ, R) , no qual [Pellegrini and Wainer, 2007]:

- S é o conjunto de estados do sistema.
- A é o conjunto de ações que podem ser executadas.
- $\Gamma : S \times A \times S \mapsto [0, 1]$ é uma função que fornece a probabilidade do sistema passar para o estado $s' \in S$, dado que a ação $a \in A$ foi executada no estado $s \in S$. A probabilidade de transição Γ é denotada por $p(s'|a,s)$.
- $R : S \mapsto \mathfrak{R}$ é uma função que fornece a recompensa para cada estado $s \in S$. A recompensa é denotada por $R(s)$.

O modelo gráfico do PDM é mostrado na Figura 4.1. Neste modelo, o estado s_t depende de s_{t-1} e a_{t-1} , e esta dependência quantifica a probabilidade de transição $p(s_t|a_{t-1}, s_{t-1})$. De forma análoga, $R(s_t)$ é dada em função de s_t .

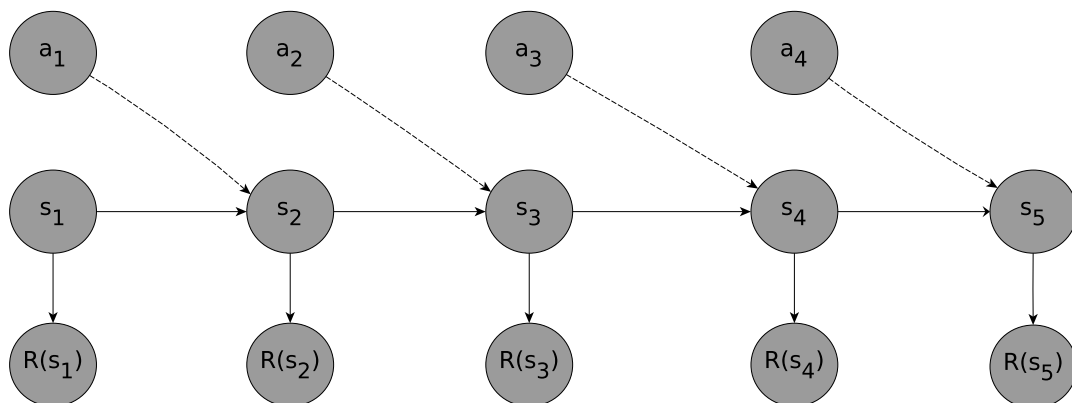


Figura 4.1 – Modelo gráfico simplificado para o PDM [Sucar, 2011].

O modelo da Figura 4.1 é simplificado, pois mostra que apenas uma ação pode ser executada em cada estado. No entanto, há um conjunto de ações A_s possíveis de serem executadas no estado $s \in S$. Assim, o conjunto de todas as ações possíveis é $A = \bigcup_{s \in S} A_s$ [Pellegrini and Wainer, 2007].

Neste capítulo vamos utilizar ações e estados finitos. Podemos dizer que um conjunto é discreto se os seus elementos são finitos ou contáveis. Portanto, o PDM também será discreto, uma vez que os estados e ações são conjuntos discretos [Feinberg and Schwartz, 2012]. A seguir, são definidos os conceitos de **objetivo**, **política**, **horizonte de planejamento** e **função valor**, termos comumente presentes em algoritmos de solução de PDM's.

4.1.1 Conceitos

Geralmente, os robôs escolhem ações que o levem até um **objetivo** (do inglês, *goal*). Os objetivos referem-se às configurações do espaço de trabalho. Por exemplo, para um robô móvel terrestre um objetivo pode ser uma pose $(x, y, \theta)^T$. Já para um manipulador pode ser a posição final $(x, y, z)^T$ da garra.

Horizonte de planejamento é o número de épocas de decisão (T) disponível para se tomar decisões. Em algumas situações basta escolher uma ação para maximizar a recompensa esperada imediata, já em outras é apropriado escolher ações cujo somatório das recompensas futuras seja máximo [Pellegrini and Wainer, 2007]. Então, é definido o termo recompensa total esperada [Thrun et al., 2005]:

$$V_T = E \left[\sum_{i=1}^T \eta^i R_{t+i} \right]. \quad (4.1)$$

A esperança $E[\cdot]$ é tomada sobre valores de recompensas futuras que o robô pode acumular entre o tempo t e $t + T$. A recompensa total esperada também é chamada de valor para diferenciá-la da recompensa imediata $R(s)$. A variável η é chamada de fator de desconto e pertence ao intervalo $[0, 1]$. O horizonte pode ser dividido em três casos:

- $T = 1$. Este é o caso “ganancioso”, no qual o robô só procura maximizar a recompensa esperada imediata. Por ser uma otimização mais simples (do que a realizada em várias etapas), pode ser aplicada para resolver problemas robóticos com baixo custo computacional. A otimização “gananciosa” é invariante com respeito à variável η , mas requer $\eta > 0$.
- T é finito (porém, maior que 1). Normalmente, a recompensa não é descontada e utiliza-se $\eta = 1$. Na otimização usando horizonte finito, a ação de controle ótima é uma função do horizonte de tempo. Isto pode acarretar em políticas ótimas diferentes para épocas de decisão próximas de T e distantes de T , o que implica em manter planos distintos para horizontes diferentes, adicionando complexidade ao problema.
- T é infinito. Neste caso, o número de épocas de decisão restantes é igual para qualquer ponto no tempo. Porém, o fator de desconto η é primordial para garantir a convergência de V_∞ . Assumindo que cada recompensa individual R é limitada em magnitude ($R \leq R_{max}$) e $\eta < 1$, V_∞ será finito:

$$V_\infty < R_{max} + \eta R_{max} + \eta^2 R_{max} + \eta^3 R_{max} + \dots = \frac{R_{max}}{1 - \eta}. \quad (4.2)$$

Assim, a otimização com horizonte finito pode ser mais difícil do que a otimização com horizonte infinito descontado.

A **política**⁹ $\pi : S \mapsto A$ é uma função que define o comportamento do agente, mapeando todos os estados $s \in S$ em ações de controle $a \in A$. A **função valor** mensura o valor esperado

⁹ A Equação (4.1) pode ser usada para comparar duas ou mais políticas.

de uma política específica [Thrun et al., 2005], e é dada pela função $V : S \mapsto \mathfrak{R}$. Por exemplo, para o horizonte de planeamento $T = 1$, temos a política que maximiza a recompensa esperada imediata. A política ótima (para $T = 1$) é obtida pelo argumento que maximiza a recompensa esperada de todas as ações a :

$$\pi_1(s) = \eta \operatorname{argmax}_a \left[\sum_{s'} R(s') p(s'|a, s) \right], \quad (4.3)$$

e a função valor associada é a recompensa esperada imediata:

$$V_1(s) = \eta \max_a \left[\sum_{s'} R(s') p(s'|a, s) \right]. \quad (4.4)$$

A partir de $V_1(s)$, podemos determinar os valores para as demais épocas de decisão de forma recursiva. A política ótima e a sua função valor para horizonte $T = 2$ são:

$$\pi_2(s) = \eta \operatorname{argmax}_a \left[\sum_{s'} [R(s') + V_1(s')] p(s'|a, s) \right], \quad (4.5)$$

$$V_2(s) = \eta \max_a \left[\sum_{s'} [R(s') + V_1(s')] p(s'|a, s) \right]. \quad (4.6)$$

Logo, para o caso de horizonte finito T teremos:

$$\pi_T(s) = \eta \operatorname{argmax}_a \left[\sum_{s'} [R(s') + V_{T-1}(s')] p(s'|a, s) \right], \quad (4.7)$$

$$V_T(s) = \eta \max_a \left[\sum_{s'} [R(s') + V_{T-1}(s')] p(s'|a, s) \right]. \quad (4.8)$$

Para o caso de horizonte infinito, a função valor ótima tende a alcançar o equilíbrio:

$$V_\infty(s) = \eta \max_a \left[\sum_{s'} [R(s') + V_\infty(s')] p(s'|a, s) \right]. \quad (4.9)$$

A Equação (4.9) é conhecida como Equação de Bellman Estocástica. Segundo Thrun et al. [2005], toda função valor V que satisfaz a condição da Equação (4.9) tem uma política induzida ótima. Em suma, no PDM a escolha de uma ação a em um estado s determina o estado s' na próxima época de decisão por meio da função de transição de estados, e produz uma recompensa $R(s')$. Concomitantemente, o tomador de decisão busca escolher políticas que sejam ótimas, considerando as épocas futuras de decisão [Puterman, 2014].

A seguir apresentaremos o algoritmo de iteração de valores, forma clássica de resolução de PDM's. Outros métodos, como o iteração de políticas, programação linear e RTDP (“*Real Time Dynamic Programming*”) podem ser vistos com mais detalhes em Pellegrini and Wainer [2007] e Puterman [2014].

4.1.2 Algoritmo de iteração de valores

O algoritmo de iteração de valores encontra políticas de controle ótimas para sistemas estocásticos. Para tal, ele aproxima a função valor (a aproximação de V é denotada por \hat{V}) como definido na Equação (4.10). O algoritmo de iteração de valores é inicializado com $\hat{V}(s) = 0 \forall s$, e atualiza a aproximação de forma recursiva [Thrun et al., 2005].

$$\hat{V}(s) = \eta \max_a \left[\sum_{s'} [R(s') + \hat{V}(s')] p(s'|a, s) \right]. \quad (4.10)$$

O Algoritmo 7 apresenta de forma genérica a iteração de valores para PDM's, e irá convergir se $\eta < 1$. A ordem que os estados são atualizados é irrelevante, desde que cada um seja atualizado “infinitamente”. No entanto, o algoritmo costuma convergir com um pequeno número de iterações para sistemas com poucos estados.

Algoritmo 7: ITERAÇÃO DE VALORES PARA PDM

Entrada: PDM (S, A, T, R)

Saída: Política π^* e função valor V^*

```

1 inicio
2   Para todo  $s \in S$  faça
3      $\hat{V}_0(s) = 0$ 
4   fim
5    $i \leftarrow 1$ 
6   Enquanto  $|\hat{V}_i(s) - \hat{V}_{i-1}(s)| \geq \varepsilon \forall s \in S$  faça
7     Para todo  $s \in S$  faça
8        $\hat{V}_i(s) = \eta \max_a [\sum_{s'} [R(s') + \hat{V}_{i-1}(s')] p(s'|a, s)]$ 
9     fim
10     $i \leftarrow i + 1$ 
11  fim-enquanto
12  Para todo  $s \in S$  faça
13     $\pi^*(s) = \eta \operatorname{argmax}_a [\sum_{s'} [R(s') + \hat{V}_i(s')] p(s'|a, s)]$ 
14  fim
15   $V^* \leftarrow \hat{V}_i(s)$ 
16  retorna  $V^*, \pi^*$ 
17 fin

```

No Algoritmo 7, as iterações se encerram quando a maior diferença entre $|\hat{V}_i(s) - \hat{V}_{i-1}(s)|$ em todos os estados $s \in S$ é menor que algum limiar ε [Pellegrini and Wainer, 2007; Sucar, 2011]. Então, o valor da política “gananciosa” \hat{V}_i não difere de V^* por mais que $2\varepsilon\eta/(1-\eta)$ para qualquer estado, isto é [Kaelbling et al., 1998]:

$$\max_{s \in S} |\hat{V}_i(s) - V^*(s)| \leq 2\varepsilon \frac{\eta}{1-\eta}. \quad (4.11)$$

Quando o algoritmo convergir, a política que é “gananciosa” em relação à função valor final $\hat{V}_i(s)$ será considerada a ótima [Thrun et al., 2005].

4.2 Processo de Decisão de Markov Parcialmente Observável (PDMPO)

Na Seção 4.1, discutimos o problema de tomada de decisão em ambientes totalmente observáveis. Nesta seção, apresentaremos o Processo de Decisão de Markov Parcialmente Observável (PDMPO), uma generalização do PDM que considera as incertezas nos efeitos das ações de controle e nas medições. O termo parcialmente observável indica que as medições são incompletas e/ou projeções ruidosas dos estados do sistema [Krishnamurthy, 2016; Pellegrini and Wainer, 2007; Thrun et al., 2005].

O PDMPO pode ser definido pela tupla $(S, A, \Gamma, R, \Omega, O)$. Os termos (S, A, Γ, R) foram explanados na Seção 4.1, e os termos (Ω, O) são descritos abaixo [Pellegrini and Wainer, 2007]:

- O é um conjunto de observações o obtidas nas épocas de decisão.
- $\Omega : S \times O \mapsto [0, 1]$ é uma função que fornece a probabilidade de uma observação o ser verificada, dado o estado s . A probabilidade de observação é denotada por $p(o|s)$.

No PDMPO, tanto o modelo perceptivo $p(o|s)$ quanto o modelo de transição $p(s'|s, a)$ são estocásticos. A representação gráfica do PDMPO é exibida na Figura 4.2. Esta representação é uma extensão da Figura 4.1, na qual cada ação tem como resultado uma observação o que é probabilisticamente relacionada aos estados do sistema. No PDMPO, as transições entre estados são probabilísticas; há possibilidade de interferir no processo executando ações e existe incerteza quanto ao resultado da ação executada.

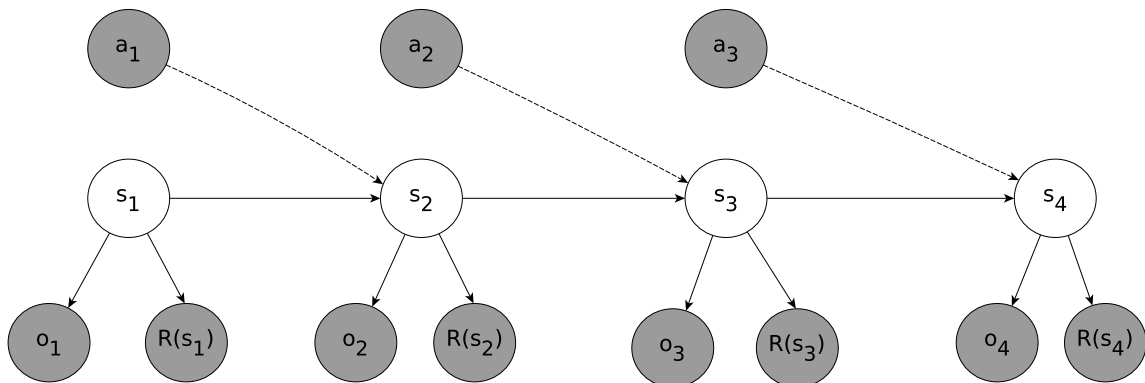


Figura 4.2 – Representação gráfica para o PDMPO [Sucar, 2011].

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 $\sum_s Bel(s) = 1$. Na Seção 3.2, utilizamos a notação $bel(x)$ para representar a

crença no espaço de estados contínuo (parametrizada por uma distribuição gaussiana multivariável). Portanto, para diferenciarmos os dois domínios indicamos a crença discreta por $Bel(s)$. 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 da variável α obtida no procedimento *forward* do ES-MOM. Conforme apresentamos no Capítulo 3, esta variável consegue estimar a localização de um robô móvel em ambientes incertos com robustez.

A definição de política também pode ser estendida para o PDMPO, ou seja, a política é o mapeamento da crença de estados $Bel(s)$ em uma ação a , e é definida pela função $\pi : \Upsilon \mapsto A$, no qual $\Upsilon = \left\{ Bel \in \mathfrak{R}^{|S|} \mid \sum_s Bel(s) = 1, Bel(s) \geq 0 \right\}$ é o espaço de todas as crenças $Bel(s)$.

No PDMPO os estados não são diretamente observáveis, assim substituímos o estado s nas Equações (4.7) e (4.8) pela crença de estados $Bel(s)$:

$$\pi_T(Bel) = \eta \operatorname{argmax}_a \left[\sum_{Bel'} [R(Bel') + V_{T-1}(Bel')] p(Bel'|a, Bel) \right], \quad (4.12)$$

$$V_T(Bel) = \eta \max_a \left[\sum_{Bel'} [R(Bel') + V_{T-1}(Bel')] p(Bel'|a, Bel) \right]. \quad (4.13)$$

O cálculo da função valor pode ser obtido de maneira relativamente simples no espaço de estados, como foi mostrado na Seção 4.1. Entretanto, o cálculo da função valor no espaço de crenças 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 Bel será contínuo e de dimensão $S - 1$, dificultando o cálculo de V_T . Outro fator de complicação é o somatório presente nas Equações (4.12) e (4.13) que é realizado sobre o espaço de crenças, ou seja, uma tarefa muito onerosa [Thrun et al., 2005].

O problema de encontrar uma função valor ótima para um PDMPO com horizonte finito exige alto custo computacional [Papadimitriou and Tsitsiklis, 1987], e para horizonte infinito é indecidível, isto é, não existe garantia que a função valor será ótima [Madani et al., 1999]. Existem alguns métodos para resolver o PDMPO, conforme mostram Brazianus [2003], Aberdeen [2003] e Murphy [2000a]. No entanto, estes métodos não são triviais, possuem alta complexidade computacional e têm aplicação limitada a problemas muito pequenos.

Diante da intratabilidade destes métodos, optamos resolver o PDMPO usando a otimização “gananciosa” (horizonte $T = 1$) para todas as épocas de decisão. Deste modo, calcularemos uma política e uma função valor para cada época de decisão:

$$\pi(Bel) = \eta \operatorname{argmax}_a \sum_{Bel'} [R(Bel') p(Bel'|a, Bel)], \quad (4.14)$$

$$V(Bel) = \eta \max_a \left[\sum_{Bel'} R(Bel') p(Bel'|a, Bel) \right]. \quad (4.15)$$

A probabilidade de transição $p(Bel'|a, Bel)$, presente na Equação (4.15), é uma distribuição sobre as crenças Bel' dado uma crença Bel e uma ação a . Se apenas Bel e a são conhecidos, Bel' não será única e $p(Bel'|a, Bel)$ será uma distribuição sobre todas as crenças. Entretanto, se conhecermos a observação o' após a execução de a , a crença posterior Bel' será única e $p(Bel'|a, Bel)$ passa a ser uma distribuição discreta degenerada, ou seja, poderá assumir apenas um único valor. Logo, concluímos que se o' também for conhecida, o somatório sobre as crenças Bel' da Equação (4.15) se tornará desnecessário. Utilizando Bel , a e o' podemos determinar Bel' por meio do filtro de Bayes. A expressão $p(Bel'|a, Bel)$ pode ser reescrita [Thrun et al., 2005]:

$$p(Bel'|a, Bel) = \sum_{o'} p(Bel'|a, Bel, o') p(o'|a, Bel), \quad (4.16)$$

sendo 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' . Adequando a Equação (4.15) temos:

$$V(Bel) = \eta \max_a \left[\sum_{Bel'} R(Bel') \sum_{o'} p(Bel'|a, Bel, o') p(o'|a, Bel) \right]. \quad (4.17)$$

$$V(Bel) = \eta \max_a \left[\sum_{o'} \underbrace{\left[\sum_{Bel'} R(Bel') p(Bel'|a, Bel, o') \right]}_{(*)} p(o'|a, Bel) \right]. \quad (4.18)$$

O somatório interno $(*)$ terá apenas um termo diferente de zero, quando isto acontece a distribuição Bel' é denotada por $C(Bel, a, o')$:

$$C(Bel, a, o')(s') = \frac{1}{p(o'|a, Bel)} p(o'|s') \sum_s p(s'|a, s) Bel(s). \quad (4.19)$$

Os passos para dedução da Equação (4.19) podem ser vistos com detalhes em Thrun et al. [2005]. Com posse da Equação (4.19), reescrevemos a Equação (4.18):

$$V(Bel) = \eta \max_a \left[\sum_{o'} R(C(Bel, a, o')) p(o'|a, Bel) \right]. \quad (4.20)$$

De forma genérica, podemos determinar a recompensa em função da crença de estados:

$$R(Bel) = \sum_{s'} R(s') Bel(s'). \quad (4.21)$$

Assim, a recompensa em função da crença $C(Bel, a, o')$ é dada por:

$$R(C(Bel, a, o')) = \sum_{s'} R(s') C(Bel, a, o')(s'). \quad (4.22)$$

$$R(C(Bel, a, o')) = \sum_{s'} R(s') \frac{1}{p(o'|a, Bel)} p(o'|s') \sum_s p(s'|a, s) Bel(s). \quad (4.23)$$

Então, substituindo a Equação (4.23) na Equação (4.20) temos:

$$V(Bel) = \eta \max_a \left[\sum_{o'} \sum_{s'} R(s') p(o'|s') \sum_s p(s'|a,s) Bel(s) \right]. \quad (4.24)$$

Comparando as Equações (4.15) e (4.24), notamos que o somatório que era realizado no espaço de crenças passou a ser computado de maneira mais eficiente no espaço de estados e observações. Também podemos dividir a maximização sobre as ações:

$$V_t(Bel_t, a) = \sum_{o'} \sum_{s'} R(s') p(o'|s') \sum_s p(s'|a,s) Bel_t(s). \quad (4.25)$$

$$V_t^*(Bel_t) = \eta \max_a V_t(Bel_t, a). \quad (4.26)$$

Nas Equações (4.25) e (4.26) incluímos a notação de tempo t , referindo-se à época de decisão t . 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 imediata pode ser determinada:

$$\pi_t^*(Bel_t) = \eta \operatorname{argmax}_a V_t(Bel_t, a). \quad (4.27)$$

Apresentamos na Figura 4.3 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 crença sobre os estados $Bel(s)$ para recomendar uma ação que maximiza a recompensa esperada do tomador de decisões (vide Equação (4.27)). No PDMPO a crença $Bel(s)$ é atualizada pelo ES-MOM, e é usada como entrada para nova política π . O efeito da execução da política no ambiente é retornado por meio da nova observação o , que será utilizada no ES-MOM. Este processo se repete iterativamente [Pellegrini and Wainer, 2007].

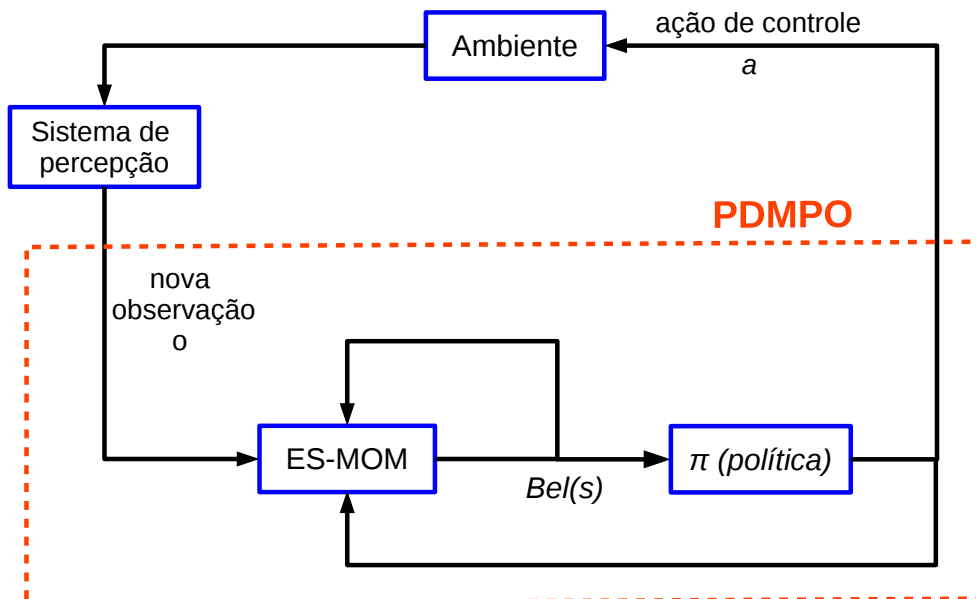


Figura 4.3 – Esquemático de funcionamento do PDMPO utilizado neste trabalho.

Utilizando as Equações (4.25), (4.26) e (4.27), desenvolvemos o Algoritmo 8 para resolver o PDMPO de forma iterativa usando a otimização “gananciosa”.

Algoritmo 8: RESOLUÇÃO DO PDMPO USANDO OTIMIZAÇÃO “GANANCIOSA”

Entrada: PDMPO $(S, A, \Gamma, R, \Omega, O)$

Estado alvo

Crença inicial sobre os estados Bel_0

Saída: Política π^* e função valor V^* (em cada época de decisão)

```

1 inicio
2   Para todo  $a \in A$  faça
3      $h \leftarrow 0$ 
4     Para todo  $o' \in O$  faça
5        $g \leftarrow 0$ 
6       Para todo  $s' \in S$  faça
7          $f \leftarrow 0$ 
8         Para todo  $s \in S$  faça
9           Se  $a \in A_s$  então
10             $f \leftarrow f + p(s'|a, s)Bel(s)$ 
11          fim
12        fim
13         $g \leftarrow g + R(s')p(o'|s')f$ 
14      fim
15       $h \leftarrow h + g$ 
16    fim
17     $V(Bel, a) \leftarrow h$ 
18  fim
19   $\pi^*(Bel) \leftarrow \eta \operatorname{argmax}_a V(Bel, a)$ 
20   $V^*(Bel) \leftarrow \eta \max_a V(Bel, a)$ 
21  executar  $\pi^*$ 
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 fin

```

O Algoritmo 8 recebe como entrada o PDMPO $(S, A, \Gamma, R, \Omega, O)$, o estado marcado como alvo, 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, verificamos a possibilidade do sistema passar para o estado s' quando a ação a é executada, da linha 6 à 14 calculamos 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, realizamos o processo acima $\forall a \in A$ e $\forall o' \in O$. Na linha 19, computamos 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 $Bel(estado_{alvo}) > 0.7$, consideramos que o robô chegou ao alvo, e o algoritmo encerrará retornando *sucesso* (linha 24). Se o estado alcançado não for o alvo ($Bel(estado_{alvo}) \leq 0.7$), o algoritmo retorna à linha 1 e repete o processo descrito acima até alcançar o estado desejado.

4.3 Simulação de controle de um robô em um galpão de estoque de produtos usando o PDMPO

Nesta seção, simulamos um sistema de controle das ações de um robô móvel terrestre em um ambiente definido topologicamente. O objetivo do sistema desenvolvido é estimar em qual estado do ambiente o robô está localizado, e a partir da estimativa selecionar uma ação de controle que o direcione a um estado marcado como alvo. Para tal, fazemos uso do Algoritmo 8, que retorna ações com recompensas (esperadas) imediatas maximizadas. Uma perspectiva do sistema é mostrada na Figura 4.4. O sistema representa um galpão utilizado para estoque de oito pilhas de produtos (os produtos podem ser dos mais diversos tipos, e as pilhas são simbolizadas com os blocos cinzas). O ambiente simulado possui 17x13m de dimensão, e um robô diferencial *Pioneer 3-DX* equipado com uma câmera Kinect se locomove por ele. Durante a movimentação o robô observa cilindros coloridos, usados para estimar sua localização.

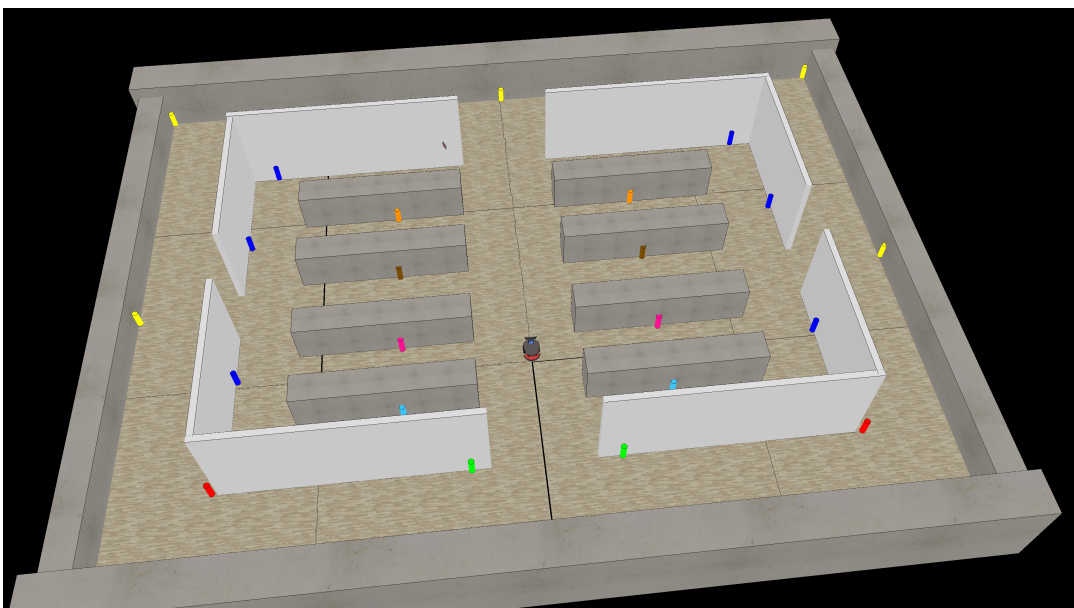


Figura 4.4 – Sistema de estocagem proposto, desenvolvido no *software* V-REP.

O sistema de estocagem foi construído no simulador **V-REP**, um ambiente baseado em arquitetura de controle distribuído, no qual cada objeto pode ser controlado individualmente por um *script* interno (em linguagem LUA) ou externo (um nó do ROS, por exemplo). Nesta aplicação, foram criados dois nós do ROS, um é o *script* em C++ e o outro é o próprio V-REP. Para estabelecer a comunicação entre os nós foram gerados quatro tópicos: *cmd_vel*, *imagem RGB*, *ângulo do Kinect* e *odometria*, que podem ser vistos na Figura 4.5. No *script* em C++ comanda-se o sistema, realizando o processamento das imagens provenientes do V-REP e enviando os comandos de velocidades para o robô (após realizar o planejamento de movimento). O V-REP simbolizando o “mundo real” envia as medições da câmera e de odometria para o *script* em C++. A câmera Kinect tem um *range* de 57° de cobertura, assim é necessário que sejam enviados ângulos de giro para o Kinect a fim de cobrir todo o espaço ao redor do robô.

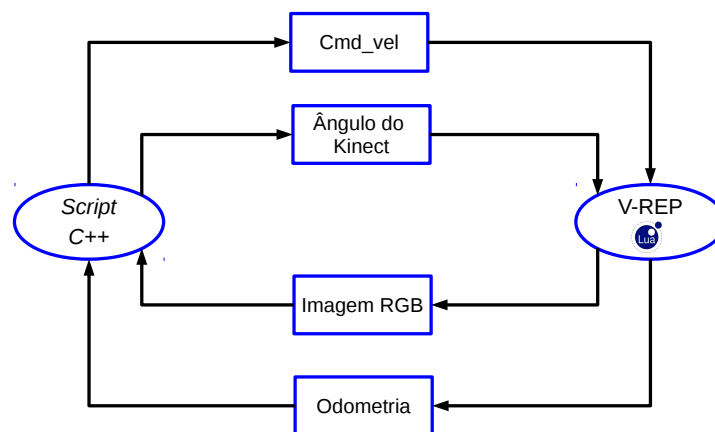


Figura 4.5 – Integração do sistema desenvolvido utilizando o *framework* ROS.

4.3.1 Mapa Topológico

Uma abstração topológica do sistema de estocagem é exibida na Figura 4.6.

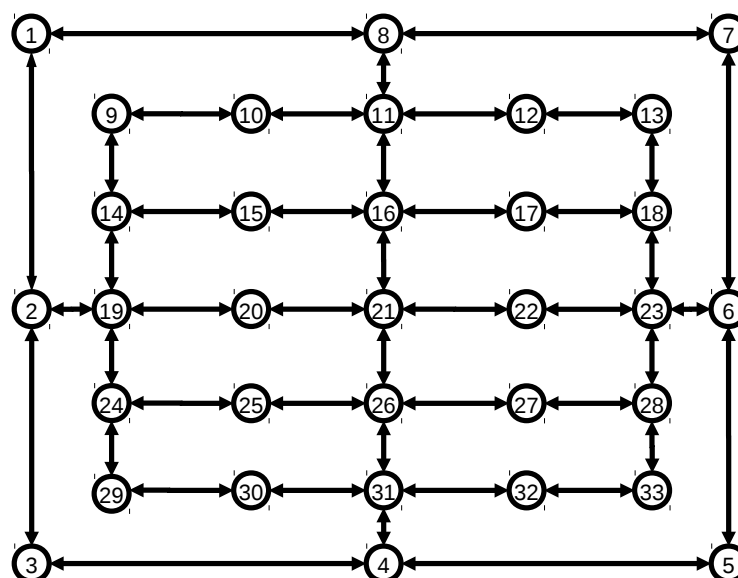


Figura 4.6 – Mapa topológico do sistema proposto.

O mapa topológico é construído *a priori* e armazenado em um banco de dados. A partir destes dados, são determinadas as posições dos 33 estados no ambiente, as conexões entre eles e conseqüentemente as 88 ações factíveis. Os estados e as arestas que os conectam estão em configurações livres do ambiente. Os estados também são associados com as suas respectivas percepções, como mostra a Tabela 4.1.

Tabela 4.1 – Observações referentes a cada estado.

Observações (estados)	o_1 (1,3, 5,7)	o_2 (2,4, 6,8)	o_3 (9,13, 29,33)	o_4 (10,12, 30,32)	o_5 (11,31)	o_6 (14,18, 24,28)	o_7 (15,17, 25,27)	o_8 (16,26)	o_9 (19,23)	o_{10} (20,22)	o_{11} (21)
Amarela	1	1	0	0	1	0	0	0	1	0	0
Azul	0	0	2	1	0	2	0	0	2	0	0
Ciano	0	0	1	1	2	0	0	0	0	0	0
Laranja	0	0	0	0	0	1	1	2	0	0	0
Marrom	0	0	0	0	0	0	0	0	2	2	4
Rosa	0	0	0	0	0	1	1	2	0	0	0
Vermelha	1	0	0	0	0	0	0	0	0	0	0
Verde	0	1	0	0	0	0	0	0	0	0	0

Ao todo, há 11 tipos de observações possíveis (o_1 a o_{11}), compostas de cilindros de oito cores: amarela, azul, ciano, laranja, marrom, rosa, vermelha e verde. Por exemplo, a observação o_1 é constituída de um cilindro amarelo e um vermelho, e pode ser vista nos estados $s_1 - s_3 - s_5 - s_7$. Semelhante aos problemas mostrados na Seção 3.3.3, o presente sistema é simétrico e ambíguo, requisitando assim de um estimador de localização robusto, para que a seleção de uma política de controle não resulte em colisão com o ambiente.

Para identificar os cilindros, convertemos as imagens capturadas do ambiente (no modelo RGB) para o modelo HSV e realizamos a calibração das oito cores dos cilindros (vide Seção 2.4). Após aplicar o filtro de cores HSV, utilizamos o filtro *canny* para realçar as componentes da imagem, e na seguida o comando *findcontours* para encontrar contornos na imagem. Desta maneira, o processamento de imagem retornará a quantidade de cilindros (de cada cor) presentes nas imagens capturadas.

4.3.2 Resultados e discussões

Os resultados da aplicação do Algoritmo 8, no sistema de estocagem simulado no V-REP, serão apresentados nesta seção. Para tal, definimos a recompensa dos estados $s' \in S$:

$$R(s') = \frac{1}{1 + D_{s'}}. \quad (4.28)$$

A função $R(s')$ incentiva o PDMPO a seguir o caminho com menor distância até o alvo. A variável $D_{s'}$ representa a distância ótima do estado s' ao estado alvo, e é obtida pela implementação do algoritmo de Dijkstra. Este algoritmo calcula o caminho ótimo de um estado

“alvo” aos demais estados de um grafo, de acordo com algum critério de otimalidade; neste trabalho o critério escolhido foi a distância euclidiana. Os demais parâmetros do Algoritmo 8 (probabilidade de transição $p(s'|a,s)$ e de observabilidade $p(o'|s')$) foram determinados de forma análoga aos testes do Capítulo 3. Ou seja, a probabilidade de transição foi calculada de acordo com a Equação (3.59), e a probabilidade de observabilidade com a Equação (3.58).

4.3.2.1 Teste 1

Para o primeiro teste, consideramos que havia uma confiança de 100% do robô estar no estado s_3 , e que o mesmo tinha como destino o estado s_{13} . A partir da informação do estado desejado atualizamos a recompensa $R(s')$. O resultado da seleção de ações com maior função valor $V(Bel,a)$ é mostrado na Tabela 4.2. Esta tabela apresenta os três maiores valores de $V(Bel,a)$ em cada época de decisão t . Como em $t = 1$, a ação $a_{3,4}$ (ação que leva o robô do estado s_3 para o s_4) 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 demais épocas de decisão, até que o critério de parada do Algoritmo 8 ($Bel(s_{13}) > 0.7$) seja atingido.

Tabela 4.2 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (usando a Equação (4.28)).

t	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$
1	$a_{3,4} \rightarrow 0.52$	$a_{3,2} \rightarrow 0.48$	-
2	$a_{4,5} \rightarrow 0.43$	$a_{4,31} \rightarrow 0.36$	$a_{4,3} \rightarrow 0.21$
3	$a_{5,6} \rightarrow 0.72$	$a_{5,4} \rightarrow 0.28$	-
4	$a_{6,23} \rightarrow 0.60$	$a_{6,7} \rightarrow 0.20$	$a_{6,5} \rightarrow 0.20$
5	$a_{23,18} \rightarrow 0.55$	$a_{23,6} \rightarrow 0.19$	$a_{23,28} \rightarrow 0.15$
6	$a_{18,13} \rightarrow 0.76$	$a_{18,23} \rightarrow 0.13$	$a_{18,17} \rightarrow 0.11$

Pela Tabela 4.2 notamos que o robô conseguiu atingir o estado desejado, porém o caminho executado não foi o ótimo. Isto acontece, pois recompensa $R(s')$ definida na Equação (4.28) considera apenas o ganho do estado futuro s' e não inclui a distância que o robô deverá percorrer ao executar a ação $a_{s,s'}$. Assim, reescrevemos a Equação (4.28) considerando a distância euclidiana $d_{s,s'}$ entre os estados s e s' :

$$R(s') = \frac{1}{1 + D_{s'} + d_{s,s'}}. \quad (4.29)$$

Fazendo uso da Equação (4.29), o primeiro teste é repetido e o resultado é mostrado na Tabela 4.3. Um vídeo ilustrativo desta simulação pode ser visto em <https://youtu.be/UatyVJstfZo>. Analisando as Tabelas 4.2 e 4.3, percebemos que os caminhos percorridos pelo robô são diferentes. A distância total percorrida na Tabela 4.2 foi de 28.2m e na Tabela 4.3 de 24.2m. Utilizando a Equação (4.29), o caminho fornecido pelo Algoritmo 8 foi um dos ótimos. Na Tabela 4.3, temos duas épocas de decisão ($t = 3, 4$) que há empate entre duas funções valores $V(Bel,a)$, nestes casos qualquer uma das ações escolhidas resultará em um caminho ótimo. Quando houver empates, executamos a ação relacionada ao primeiro valor do ordenamento de $V(Bel,a)$. Nos demais testes do capítulo consideramos a recompensa $R(s')$ da Equação (4.29).

Tabela 4.3 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (usando a Equação (4.29)).

t	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$
1	$a_{3,2} \rightarrow 0.51$	$a_{3,4} \rightarrow 0.49$	-
2	$a_{2,19} \rightarrow 0.48$	$a_{2,1} \rightarrow 0.30$	$a_{2,3} \rightarrow 0.22$
3	$a_{19,14} \rightarrow 0.30$	$a_{19,20} \rightarrow 0.30$	$a_{19,2} \rightarrow 0.22$
4	$a_{14,9} \rightarrow 0.36$	$a_{14,15} \rightarrow 0.36$	$a_{14,19} \rightarrow 0.28$
5	$a_{9,10} \rightarrow 0.60$	$a_{9,14} \rightarrow 0.40$	-
6	$a_{10,11} \rightarrow 0.65$	$a_{10,9} \rightarrow 0.35$	-
7	$a_{11,12} \rightarrow 0.50$	$a_{11,8} \rightarrow 0.20$	$a_{11,16} \rightarrow 0.16$
8	$a_{12,13} \rightarrow 0.70$	$a_{12,11} \rightarrow 0.30$	-

Na Tabela 4.3, consideramos que a confiança inicial era de 100% no estado s_3 , e o ES-MOM manteve o rastreamento da crença em 100% no verdadeiro estado do robô para as demais épocas de decisão. Já na Tabela 4.4, mostramos o resultado para uma crença inicial de 30% do robô estar no estado s_3 .

Tabela 4.4 – Ações selecionadas entre os estados s_3 e s_{13} , para cada época de decisão t (considerando uma crença inicial de 30% do robô estar no estado s_3).

t	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$
1	$a_{3,2} \rightarrow 0.12$	$a_{3,4} \rightarrow 0.11$	$a_{18,13} \rightarrow 0.04$
2	$a_{2,19} \rightarrow 0.25$	$a_{2,1} \rightarrow 0.20$	$a_{6,23} \rightarrow 0.17$
3	$a_{19,14} \rightarrow 0.30$	$a_{19,20} \rightarrow 0.30$	$a_{19,2} \rightarrow 0.22$
4	$a_{14,9} \rightarrow 0.36$	$a_{14,15} \rightarrow 0.36$	$a_{14,19} \rightarrow 0.28$
5	$a_{9,10} \rightarrow 0.60$	$a_{9,14} \rightarrow 0.40$	-
6	$a_{10,11} \rightarrow 0.65$	$a_{10,9} \rightarrow 0.35$	-
7	$a_{11,12} \rightarrow 0.50$	$a_{11,8} \rightarrow 0.20$	$a_{11,16} \rightarrow 0.16$
8	$a_{12,13} \rightarrow 0.70$	$a_{12,11} \rightarrow 0.30$	-

Podemos observar na Tabela 4.4 que até o estimador ES-MOM convergir para real localização do robô (épocas de decisão $t = 1, 2$), a amplitude dos valores de $V(Bel, a)$ é menor do que na Tabela 4.3, indicando assim uma menor margem de segurança para tomar uma decisão. Após a convergência do ES-MOM ($t > 2$), os valores da Tabela 4.4 foram iguais aos da Tabela 4.3. Foram realizadas outras tentativas com menor nível de confiança inicial, inclusive com uma distribuição uniforme, entretanto o Algoritmo 8 apresentou inconsistências, resultando em colisões com o ambiente. Logo, podemos concluir que um bom funcionamento do Algoritmo 8 está condicionado a uma estimativa de localização coerente.

4.3.2.2 Teste 2

O segundo teste é semelhante ao anterior, porém passamos alguns lugares para o robô atingir em sequência. Os estados desejados foram s_{21} , s_{23} e s_8 , sendo que o robô partiu do estado s_{33} com confiança de 60%. Quando o robô atingiu cada um dos *waypoints*, ele calculou novamente a distância $D_{s'} \forall s' \in S$, tomando como referência o “novo estado alvo”. Os resultados são mostrados na Tabela 4.5.

Tabela 4.5 – Ações selecionadas entre os estados $s_{33} \rightarrow s_{21} \rightarrow s_{23} \rightarrow s_8$.

t	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$	$a_{s,s'} \rightarrow V(Bel,a)$
1	$a_{33,32} \rightarrow 0.50$	$a_{33,28} \rightarrow 0.50$	-
2	$a_{32,31} \rightarrow 0.65$	$a_{32,33} \rightarrow 0.35$	-
3	$a_{31,26} \rightarrow 0.42$	$a_{31,4} \rightarrow 0.22$	$a_{31,32} \rightarrow 0.18$
4	$a_{26,21} \rightarrow 0.50$	$a_{26,31} \rightarrow 0.21$	$a_{26,27} \rightarrow 0.15$
5	$a_{21,22} \rightarrow 0.40$	$a_{21,16} \rightarrow 0.22$	$a_{21,26} \rightarrow 0.22$
6	$a_{22,23} \rightarrow 0.72$	$a_{22,21} \rightarrow 0.28$	-
7	$a_{23,18} \rightarrow 0.33$	$a_{23,22} \rightarrow 0.33$	$a_{23,6} \rightarrow 0.20$
8	$a_{18,17} \rightarrow 0.37$	$a_{18,13} \rightarrow 0.37$	$a_{18,23} \rightarrow 0.26$
9	$a_{17,16} \rightarrow 0.65$	$a_{17,18} \rightarrow 0.35$	-
10	$a_{16,11} \rightarrow 0.42$	$a_{16,21} \rightarrow 0.22$	$a_{16,17} \rightarrow 0.18$
11	$a_{11,8} \rightarrow 0.60$	$a_{11,16} \rightarrow 0.20$	$a_{11,12} \rightarrow 0.10$

Outra aplicação possível é o robô atingir um estado desejado, e quando alcançá-lo retornar até o local inicial, simbolizando a entrega ou a busca de um produto.

4.4 Experimento em um espaço de convivência usando o PDMPO

Nesta seção, vamos apresentar experimentos reais usando o Algoritmo 8 para resolver o PDMPO. O cenário escolhido foi um espaço de convivência próximo à biblioteca da Escola de Engenharia da UFMG. Uma perspectiva do ambiente é mostrada na Figura 4.7. 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 dividido em três áreas, que são exibidas na Figura 4.8.

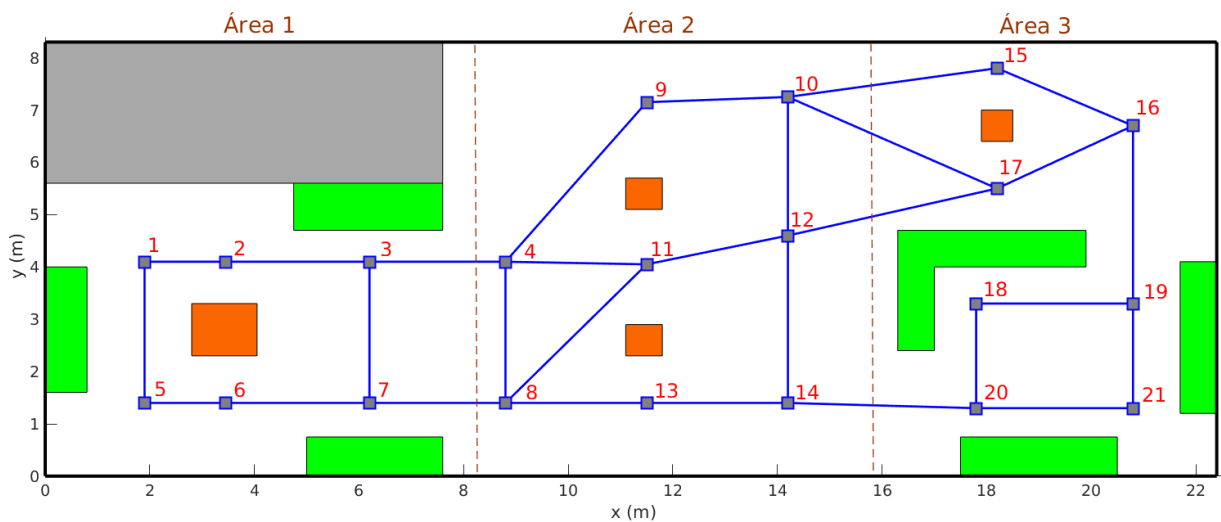


Figura 4.7 – Perspectiva do espaço de convivência utilizado nos experimentos do PDMPO. Os retângulos verdes representam sofás, os laranjas *poofs*, e o cinza uma escada. As conexões entre os 21 estados do ambiente são indicadas pelas linhas azuis.



(a) Imagem da Área 1.

(b) Imagem da Área 2.

(c) Imagem da Área 3.

Figura 4.8 – Espaço de convivência utilizado nos experimentos do PDMPO.

O robô usado nos experimentos foi a plataforma robótica apresentada na Figura 3.8. No ES-MOM, usamos as medições da câmera *Tracking* para calcular o estado \bar{s} , e as câmeras RGB's para identificar Arucos espalhados pelo ambiente, de acordo com a Seção 3.3.3.2. Próximo a cada estado do ambiente existe um Aruco (utilizados como *landmarks*) possuindo uma id que varia 1 a 10, conforme a relação apresentada na Tabela 4.6. Notamos que o ambiente é ambíguo, pois uma id pode ser vista em mais de um estado, assim a probabilidade de observabilidade é determinada seguindo a Equação (3.58). Os demais parâmetros do Algoritmo 8 são obtidos de forma análoga aos da Seção 4.3.

Tabela 4.6 – Símbolos de observações (id's) referentes a cada estado (experimento do PDMPO).

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

4.4.1 Teste 1

No primeiro teste, consideramos que havia uma confiança inicial de 80% do robô estar no estado s_5 e que o objetivo era atingir o estado s_{16} . O resultado da seleção de ações com maior função valor $V(Bel, a)$ é mostrado na Tabela 4.7. Durante o trajeto, o ES-MOM manteve uma boa estimativa de localização e o caminho executado pelo robô foi o ótimo. Um vídeo ilustrativo do experimento pode ser visto em <https://youtu.be/67psg3uXHb8>.

Tabela 4.7 – Ações selecionadas entre os estados s_5 e s_{16} , para cada época de decisão t .

t	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$
1	$a_{5,6} \rightarrow 0.54$	$a_{5,1} \rightarrow 0.46$	-
2	$a_{6,7} \rightarrow 0.59$	$a_{6,5} \rightarrow 0.41$	-
3	$a_{7,8} \rightarrow 0.39$	$a_{7,3} \rightarrow 0.33$	$a_{7,6} \rightarrow 0.28$
4	$a_{8,11} \rightarrow 0.29$	$a_{8,4} \rightarrow 0.26$	$a_{8,13} \rightarrow 0.25$
5	$a_{11,12} \rightarrow 0.49$	$a_{11,4} \rightarrow 0.28$	$a_{11,8} \rightarrow 0.23$
6	$a_{12,17} \rightarrow 0.38$	$a_{12,10} \rightarrow 0.26$	$a_{12,11} \rightarrow 0.21$
7	$a_{17,16} \rightarrow 0.48$	$a_{17,12} \rightarrow 0.26$	$a_{17,10} \rightarrow 0.26$

4.4.2 Teste 2

Um segundo teste foi realizado, e os estados desejados foram o s_{12} e o s_1 (em sequência), para tal consideramos que havia uma confiança inicial de 30% do robô estar no estado s_{21} . Os resultados são exibidos na Tabela 4.8. Podemos ver que na primeira iteração do Algoritmo 8, há pequena margem de segurança para executar a ação $a_{21,20}$ 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. O caminho executado do estado s_{21} ao estado s_1 foi o ótimo. 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 4.8 – Ações selecionadas entre os estados $s_{21} \rightarrow s_{12} \rightarrow s_1$, para cada época de decisão t .

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

4.4.3 Teste 3

No terceiro teste, consideramos que o robô devia se direcionar para o estado s_{18} , porém não havia nenhuma informação sobre a sua localização inicial. Desta forma, ele foi guiado inicialmente por um usuário utilizando um *joystick*, e conforme coletava informações das *landmarks* o ES-MOM fazia a atualização das probabilidades α '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 8. 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 o resultado da estimação da localização neste percurso é mostrado na Tabela 4.9.

Tabela 4.9 – Probabilidade α do ES-MOM para o trajeto $[s_1 - s_2 - s_3 - s_4 - s_{11} - s_{12} - s_{17}]$.

α_1	α_2	α_3	α_4	α_5	α_6	α_7
$\alpha(1)=0.50$	$\alpha(2)=0.50$	$\alpha(3)=0.50$	$\alpha(4)=0.50$	$\alpha(11)=0.50$	$\alpha(12)=0.53$	$\alpha(17)=0.91$
$\alpha(5)=0.50$	$\alpha(6)=0.50$	$\alpha(7)=0.50$	$\alpha(8)=0.50$	$\alpha(13)=0.50$	$\alpha(14)=0.45$	$\alpha(15)=0.06$
$\alpha(21)=0.00$	$\alpha(21)=0.00$	$\alpha(21)=0.00$	$\alpha(21)=0.00$	$\alpha(21)=0.00$	$\alpha(10)=0.02$	$\alpha(20)=0.02$

Pela Tabela 4.9 podemos ver que o ES-MOM atingiu uma confiança maior que 70% na sétima iteração. A partir daí, o Algoritmo 8 entrou em funcionamento e o resultado da seleção de ações é mostrado na Tabela 4.10.

Tabela 4.10 – Ações selecionadas entre os estados s_{17} e s_{18} , para cada época de decisão t .

t	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$	$a_{s,s'} \rightarrow V(Bel, a)$
1	$a_{17,16} \rightarrow 0.44$	$a_{17,12} \rightarrow 0.34$	$a_{17,10} \rightarrow 0.22$
2	$a_{16,19} \rightarrow 0.48$	$a_{16,17} \rightarrow 0.26$	$a_{16,15} \rightarrow 0.26$
3	$a_{19,18} \rightarrow 0.61$	$a_{19,21} \rightarrow 0.30$	$a_{19,16} \rightarrow 0.09$

Se o robô fosse desde o início de forma autônoma certamente aconteceria alguma colisão com o ambiente. Entretanto, usando uma aplicação semiautônoma, o robô conseguiu atingir o estado desejado de maneira segura. Um vídeo ilustrando este experimento pode ser visualizado em <https://youtu.be/YEMkbigulq0>.

4.5 Conclusões do capítulo

No presente capítulo, o PDMPO foi apresentado como uma alternativa para resolver o problema de planejamento das ações de um robô que se move por um sistema de estocagem simulado e por um espaço de convivência real. 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. O PDMPO é uma abordagem mais abrangente que o PDM, porém é mais difícil de se resolver. Os experimentos e as simulações mostram que o Algoritmo 8 apresentou sucesso na seleção das políticas de controle. Por meio da otimização “gananciosa” e das simplificações utilizadas, o algoritmo 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. Se os valores da crença inicial forem mais consistentes, a diferença entre o maior e o segundo maior valor de $V(Bel, a)$ é amplificada. Desta forma, o Algoritmo 8 se mostra adequado para fazer o rastreamento e planejamento das ações do robô até o alvo, desde que se tenha algum conhecimento (mesmo que mínimo) sobre a configuração inicial. Há de se salientar, que os caminhos executados pelo robô foram ótimos, porém podem não ser os únicos.

CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação, utilizamos os modelos de Markov para lidar com as incertezas presentes no planejamento de movimento de um robô em um mapa topológico. No Capítulo 3, tratamos das incertezas na estimação da localização por meio da estrutura perceptiva do ES-MOM. O ES-MOM realizou a inferência do estado do robô a partir de *landmarks* artificiais espalhadas pelo ambiente. O ES-MOM se mostrou robusto mesmo em ambientes com pouca distinguibilidade (que possuem ambiguidade e simetria), e sem ter conhecimento prévio sobre a localização inicial.

No Capítulo 4, integramos a localização proveniente do ES-MOM e o planejamento de movimento por meio do PDMPO. A aplicação do PDMPO resulta em políticas de controle ótimas a serem executadas pelo robô a fim de atingir um local desejado do ambiente. Neste trabalho, optamos por resolver o PDMPO usando uma otimização “gananciosa”, maximizando as recompensas imediatas em vez da recompensa acumulada. Consequentemente, as políticas são encontradas com menor complexidade computacional. Nos Capítulos 3 e 4, validamos os métodos propostos mediante simulações e experimentos com um robô real.

No Capítulo 4, utilizamos o planejador de movimento local do Apêndice A para executar as políticas de controle. Este planejador local foi responsável pela navegação do robô entre os estados do ambiente topológico. Entretanto, este planejador não considera a presença de obstáculos entre os estados, nem restrições nas ações de controle. Dadas as limitações do planejador do Apêndice A, apresentamos no Apêndice B uma alternativa para executar a política fornecida pelo PDMPO, sob restrições de controles e regiões proibidas: a Programação Dinâmica Diferencial (PDD). A função deste novo planejador local é garantir uma navegação (entre os estados) sem movimentos bruscos e sem colidir com objetos distribuídos pelo ambiente.

No Apêndice B, realizamos algumas simulações usando a PDD para gerar trajetórias ótimas (entre dois estados) sob as referidas restrições, e os resultados foram satisfatórios. Como trabalhos futuros, pretendemos combinar as técnicas discutidas no Capítulo 4 e no Apêndice B. Nesta combinação, o PDMPO seria responsável pelo planejamento de movimento global

(“ações em alto nível”) do robô, determinando a sequência de estados a serem alcançados. Já a PDD assumiria a função de realizar o planejamento local (“ações em baixo nível”), gerando uma trajetória factível de ser seguida pelo robô, quando o PDMPO determinar uma política de mudança de estados.

PLANEJADOR DE MOVIMENTO LOCAL

A navegação entre estados envolve um planejador de movimento local. Este planejador local consiste em criar rotinas de locomoção entre estados com conexões factíveis. Para um estado inicial arbitrário $x(t_0) = (x_0, y_0)^T$ e um final $x(t_f) = (x_f, y_f)^T$, o planejador encontra uma trajetória (caminho parametrizado pelo tempo) que conecta as referidas configurações. Como a trajetória é parametrizada pelo tempo, é desejado que o robô esteja em $x(t_f)$ após decorridas \top unidades de tempo. A Equação (A.1) descreve o rastreamento da posição desejada para o robô durante a navegação entre $x(t_0)$ e $x(t_f)$, para cada $t \in [0, \top]$.

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x_f - x_0 \\ y_f - y_0 \end{bmatrix} g(t). \quad (\text{A.1})$$

Na Equação (A.1) foi estabelecido o polinômio auxiliar $g(t) \in [0, 1]$, responsável pela parametrização da trajetória do robô no tempo. No estado inicial teremos $g(t_0) = 0$, e no final $g(t_f) = 1$, ou seja, $g(0) = 0$ e $g(\top) = 1$. Como o robô deverá “parar” quando atingir um estado (para capturar as informações das *landmarks*), é necessário que ele inicie o trajeto com velocidade inicial e final igual a zero. Deste modo, determinamos a condição inicial e final de velocidade $\therefore \dot{g}(0) = 0$ e $\dot{g}(\top) = 0$. Um polinômio $g(t)$ de terceira ordem pode ser usado para atender estas condições:

$$\begin{cases} g(t) = \rho_1 t^3 + \rho_2 t^2 + \rho_3 t + \rho_4 \\ \dot{g}(t) = 3\rho_1 t^2 + 2\rho_2 t + \rho_3 \end{cases} \quad \begin{matrix} g(0)=0, \dot{g}(0)=0 \\ \xrightarrow{\quad} \end{matrix} \quad \begin{cases} g(t) = \rho_1 t^3 + \rho_2 t^2 \\ \dot{g}(t) = 3\rho_1 t^2 + 2\rho_2 t \end{cases}. \quad (\text{A.2})$$

Usando as condições $g(\top) = 1$ e $\dot{g}(\top) = 0$, o sistema da Equação (A.2) pode ser resolvido:

$$\begin{cases} g(\top) = \rho_1 \top^3 + \rho_2 \top^2 = 1, \\ \dot{g}(\top) = 3\rho_1 \top^2 + 2\rho_2 \top = 0, \end{cases} \quad (\text{A.3})$$

obtendo os coeficientes $\rho_1 = -2/\top^3$ e $\rho_2 = 3/\top^2$.

A Figura A.1 apresenta o gráfico de $g(t)$ e $\dot{g}(t)$ para $\top = 10$.

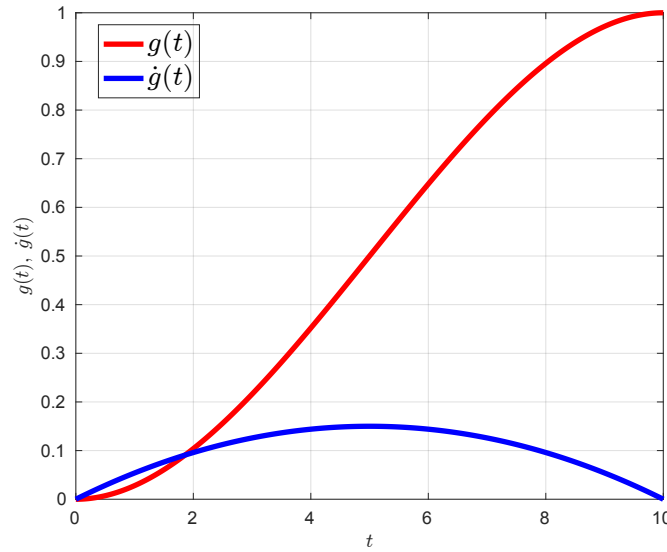


Figura A.1 – Gráfico do polinômio auxiliar $g(t)$ e sua derivada primeira $\dot{g}(t)$ para $\Upsilon = 10$.

Para um problema real, o parâmetro Υ pode ser determinado por meio da velocidade média do trajeto entre $x(t_0)$ e $x(t_f)$, conforme mostra a Equação (A.4).

$$\Upsilon = \frac{\|x(t_0) - x(t_f)\|}{v_{med}}. \quad (\text{A.4})$$

Portanto, podemos reescrever a Equação (A.1):

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x_f - x_0 \\ y_f - y_0 \end{bmatrix} \left(\frac{-2t^3}{\Upsilon^3} + \frac{3t^2}{\Upsilon^2} \right). \quad (\text{A.5})$$

Derivando a Equação (A.5) encontramos:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix} = \begin{bmatrix} x_f - x_0 \\ y_f - y_0 \end{bmatrix} \frac{dg(t)}{dt} = \begin{bmatrix} x_f - x_0 \\ y_f - y_0 \end{bmatrix} \left(\frac{-6t^2}{\Upsilon^3} + \frac{6t}{\Upsilon^2} \right). \quad (\text{A.6})$$

Os pares $(x_r, y_r)^T$ e $(\dot{x}_r, \dot{y}_r)^T$, obtidos nas Equações (A.5) e (A.6), respectivamente, são usados como referências a serem seguidas pelo controlador *feedback linearization*¹⁰. O objetivo do *feedback linearization* é controlar um ponto que está a uma distância d do centro do robô não holonômico, fornecendo as ações de controle $u_t = (v_t, \omega_t)^T$. O controlador é modelado pelas Equações (A.7) e (A.8). A pose atual do robô é denotada por $(x_t, y_t, \theta_t)^T$, e o ganho do controlador é representado por k_p .

$$\begin{bmatrix} \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix} + k_p \begin{bmatrix} x_r - x_t \\ y_r - y_t \end{bmatrix} \quad (\text{A.7})$$

$$\begin{bmatrix} v_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \cos(\theta_t) & \sin(\theta_t) \\ -\sin(\theta_t)/d & \cos(\theta_t)/d \end{bmatrix} \begin{bmatrix} \dot{x}_t \\ \dot{y}_t \end{bmatrix} \quad (\text{A.8})$$

A partir $(x_r, y_r)^T$ e $(\dot{x}_r, \dot{y}_r)^T$, o controlador garante que a posição desejada $x(t_f) = (x_f, y_f)^T$ será alcançada.

¹⁰ Mais informações sobre o controlador *feedback linearization* são apresentadas no capítulo 10 do livro “Robot modeling and control” de Spong et al. [2006].

PLANEJADOR DE MOVIMENTO LOCAL COM RESTRIÇÕES DE CONTROLE E OBSTÁCULOS

Para um robô percorrer o trajeto da configuração inicial até a configuração desejada, é necessário que aconteça o planejamento do movimento a ser realizado, de modo que o robô seja conduzido de forma segura e eficiente. Os métodos tradicionais de geração de trajetórias encontrados na literatura fazem uso da interpolação dos pontos de um caminho por meio de funções polinomiais do tempo [Charles et al., 2013], como o planejador de trajetórias apresentado no Apêndice A. A principal desvantagem da geração de trajetórias com polinômios é que a solução depende da ordem do polinômio, e o cálculo dos parâmetros do polinômio pode não ser eficiente. Além do fato de que polinômios de ordem elevada apresentam grandes oscilações.

Neste apêndice vamos apresentar a Programação Dinâmica Diferencial - PDD (do inglês, *Differential Dynamic Programming*), método eficiente de geração de trajetórias ótimas entre dois estados de um mapa topológico. A PDD é um método iterativo que a partir de uma solução inicial (ou trajetória inicial), busca encontrar melhores soluções dentro de uma vizinhança determinada por esta solução. A cada iteração da PDD, encontra-se uma melhor trajetória que é utilizada como nova solução inicial para a próxima iteração [Bacalhau et al., 2015].

À estrutura da PDD incorporaremos restrições de velocidades máximas e mínimas, e de espaços que o robô não pode ocupar, ou seja, obstáculos. Obtendo assim, um planejador local para executar as políticas do PDMPO sujeito às restrições de controle e obstáculos. Este planejador local pode ser usado de forma suplementar ao planejador do Apêndice A, que só contempla situações que não há nenhum obstáculo entre as configurações.

Desta feita, discorreremos sobre a PDD na Seção B.1. Na Seção B.2, mostraremos uma simulação usando a PDD para gerar trajetórias em um ambiente sem restrições. Na Seção B.3, incluiremos na PDD restrições de controle e obstáculos. Neste apêndice trabalharemos com estados contínuos, ações contínuas e tempo discreto.

B.1 Programação Dinâmica Diferencial

A Programação Dinâmica (PD) é uma técnica de otimização desenvolvida por [Bellman \[1957a\]](#), e pode ser aplicada para solucionar diversos problemas de controle. A programação dinâmica transforma um problema de otimização complexo em uma família de subproblemas, e para cada subproblema determina a melhor solução. A ideia básica da programação dinâmica é construir por etapas uma resposta ótima, combinando respostas já obtidas para partes menores. Desta forma, resolvendo cada subproblema parcialmente, resolve-se todo o problema através do procedimento de otimização multiestágio. O fundamento teórico para a resolução da programação dinâmica está no *princípio da otimalidade de Bellman*: “a partir de cada ponto da trajetória ótima, a trajetória restante é a ótima iniciando neste ponto”. Ou seja, em uma sequência ótima de decisões, cada subsequência também deve ser ótima. Isto é, supondo que se conheça o caminho mais curto entre o ponto A e C , e que ele passa necessariamente por B . Logo, o menor caminho entre A e C deve incluir o menor caminho entre B e C [[Bacalhau et al., 2015](#); [Baumeister and Leitão, 2014](#); [Bertsekas et al., 1995](#); [Cardoso, 2008](#)].

Utilizando a notação de função valor V discutida no Capítulo 4, podemos definir o custo ótimo (no instante de tempo t) em relação a uma configuração desejada:

$$V_t(x_t) = \min_u [l_t(x_t, u_t) + V_{t+1}(x_{t+1})], \quad (\text{B.1})$$

sendo, $l_t(x_t, u_t)$ o custo imediato e $x_{t+1} = f(x_t, u_t)$ a dinâmica que descreve a evolução do estado $x_t \in \mathfrak{R}^n$, dada uma ação de controle $u_t \in \mathfrak{R}^m$. No qual, n é a dimensão do estado x e m a dimensão do controle u . A Equação (B.1) é conhecida como Equação de Bellman Determinística.

Definindo a condição de contorno $V_T(x_T) = l_f(x_T)$, o princípio da programação dinâmica reduz a minimização ao longo de uma sequência inteira de controles para uma sequência de minimizações sobre um único controle por meio do procedimento *backward*.

No Capítulo 4, a função valor era obtida por meio da maximização, uma vez que envolve a probabilidade de observabilidade e de transição de estados. E quanto maior forem estes valores, maior é a certeza do robô se encontrar no referido estado. Neste apêndice, estamos lidando com o problema de roteamento sem considerar incertezas¹¹, portanto é conveniente substituir a “recompensa” pelo “custo” e a “maximização” pela “minimização”.

A principal desvantagem da programação dinâmica é o problema da explosão computacional, também conhecido como “maldição da dimensionalidade”. Existem algumas aproximações usadas para resolver um problema de PD, porém elas podem não ser eficientes [[Bacalhau et al., 2015](#)]. Para diminuir a complexidade do problema de PD podemos utilizar um método iterativo proposto por [Jacobson and Mayne \[1970\]](#), denominado Programação Dinâmica Diferencial (PDD). A partir de uma trajetória inicial, este método busca encontrar melhores soluções dentro de uma vizinhança determinada por esta solução inicial. A cada iteração uma melhor trajetória é

¹¹ O presente apêndice trata da programação dinâmica para o caso determinístico, enquanto o PDM apresentado na Seção 4.1 é a versão estocástica da programação dinâmica (para estados discretos).

encontrada e utilizada como a nova solução inicial (ou referência) para a próxima iteração. Para tal, a PDD utiliza a expansão quadrática do *princípio de otimalidade de Bellman* na vizinhança da trajetória de referência. Movendo iterativamente para os mínimos das aproximações quadráticas, a trajetória é melhorada progressivamente em direção a um ótimo local [Bacalhau et al., 2015; Jacobson and Mayne, 1970].

Cada iteração da PDD é composta de duas etapas: *backward* e a propagação de movimento. Na primeira é calculado o controle ótimo da trajetória de referência do sistema quadrático expandido, na segunda a trajetória de referência é atualizada usando o controle obtido na etapa *backward*. Este processo é repetido até a convergência para configuração desejada.

B.1.1 Procedimento backward

Nesta etapa, procede-se “de volta no tempo” para gerar um modelo local de V . A função valor V mostrada na Equação (B.1) pode ser aproximada por uma função Q (não minimizada):

$$Q_t(x_t, u_t) = l_t(x_t, u_t) + V_{t+1}(x_{t+1}). \quad (\text{B.2})$$

Utilizando a expansão quadrática de Taylor da função valor Q , em torno do presente par de estados (x_t, u_t) , temos:

$$Q_t(x_t, u_t) = Q_t + Q_{x,t} \delta x_t + Q_{u,t} \delta u_t + \frac{1}{2} \begin{bmatrix} \delta x_t^T & \delta u_t^T \end{bmatrix} \begin{bmatrix} Q_{xx,t} & Q_{xu,t} \\ Q_{ux,t} & Q_{uu,t} \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}. \quad (\text{B.3})$$

Os termos $\delta x_t = \hat{x}_t - x_t$ e $\delta u_t = \hat{u}_t - u_t$ representam a variação em torno de x_t e u_t , respectivamente. Sendo, \hat{x}_t e \hat{u}_t configurações pertencentes à nova trajetória (vide Equação (B.8) e (B.9)), e x_t e u_t pertencentes à trajetória de referência. Os coeficientes da expansão de Q são:

$$Q_{x,t} = l_{x,t} + V_{x,t+1} f_{x,t}, \quad (\text{B.4a})$$

$$Q_{u,t} = l_{u,t} + V_{x,t+1} f_{u,t}, \quad (\text{B.4b})$$

$$Q_{xx,t} = l_{xx,t} + V_{x,t+1} f_{xx,t} + f_{x,t}^T V_{xx,t+1} f_{x,t}, \quad (\text{B.4c})$$

$$Q_{uu,t} = l_{uu,t} + V_{x,t+1} f_{uu,t} + f_{u,t}^T V_{xx,t+1} f_{u,t}, \quad (\text{B.4d})$$

$$Q_{xu,t} = l_{xu,t} + V_{x,t+1} f_{xu,t} + f_{x,t}^T V_{xx,t+1} f_{u,t}, \quad (\text{B.4e})$$

$$Q_{ux,t} = l_{ux,t} + V_{x,t+1} f_{ux,t} + f_{u,t}^T V_{xx,t+1} f_{x,t}. \quad (\text{B.4f})$$

Além do índice de tempo t , temos o subscrito x e/ou u que indica a variável na qual a derivada é tomada. Uma vez obtido o modelo local Q , a minimização da aproximação quadrática é calculada por meio da minimização de δu_t :

$$\delta u_t^* = \underset{\delta u_t}{\operatorname{argmin}} [Q_t(x_t, u_t)] = -Q_{uu,t}^{-1} (Q_{u,t} + Q_{ux,t} \delta x_t). \quad (\text{B.5})$$

Aplicando a política δu_t^* na Equação (B.3), a aproximação quadrática de V_t é obtida:

$$V_t = Q_t - Q_{u,t}^T (Q_{uu,t})^{-1} Q_{u,t}, \quad (\text{B.6a})$$

$$V_{x,t} = Q_{x,t} - Q_{u,t}(Q_{uu,t})^{-1}Q_{ux,t}, \quad (\text{B.6b})$$

$$V_{xx,t} = Q_{xx,t} - Q_{xu,t}(Q_{uu,t})^{-1}Q_{ux,t}. \quad (\text{B.6c})$$

A etapa *backward* começa com a inicialização da função valor com o custo terminal $V_T(x_T)$, e então computa-se recursivamente as Equações (B.4), (B.5) e (B.6) para $t = T - 1, \dots, 1$.

B.1.2 Propagação de movimento

Sendo completada a etapa *backward* e assumindo que a configuração inicial x_0 é conhecida, a etapa de propagação de movimento combina as ações de controle δu_t^* para computar uma nova sequência de controle $\hat{U} = \hat{u}_0, \dots, \hat{u}_{T-1}$, e conseqüentemente uma nova trajetória $\hat{X} = \hat{x}_0, \dots, \hat{x}_T$ é gerada:

$$\hat{x}_0 = x_0, \quad (\text{B.7})$$

$$\hat{u}_t = u_t + \delta u_t^*, \quad (\text{B.8})$$

$$\hat{x}_{t+1} = f(\hat{x}_t, \hat{u}_t). \quad (\text{B.9})$$

As etapas *backward* e propagação de movimento são repetidas até a convergência.

B.2 Geração de trajetórias sem restrições

Nesta seção, vamos utilizar a PDD para geração de trajetórias ótimas para um robô móvel não holonômico, em um ambiente sem restrições de estados (obstáculos) e de entradas de controle. O modelo cinemático do robô não holonômico é o mesmo apresentado na Equação (3.52), porém sem considerar o ruído de predição.

O custo imediato l_t e final l_f foram definidos utilizando funções de custo quadráticas:

$$l_t(x_t, u_t) = (x_t - x_{goal})^T L(x_t - x_{goal}) + u_t^T W u_t, \quad (\text{B.10})$$

$$l_f(x_T) = (x_T - x_{goal})^T L(x_T - x_{goal}). \quad (\text{B.11})$$

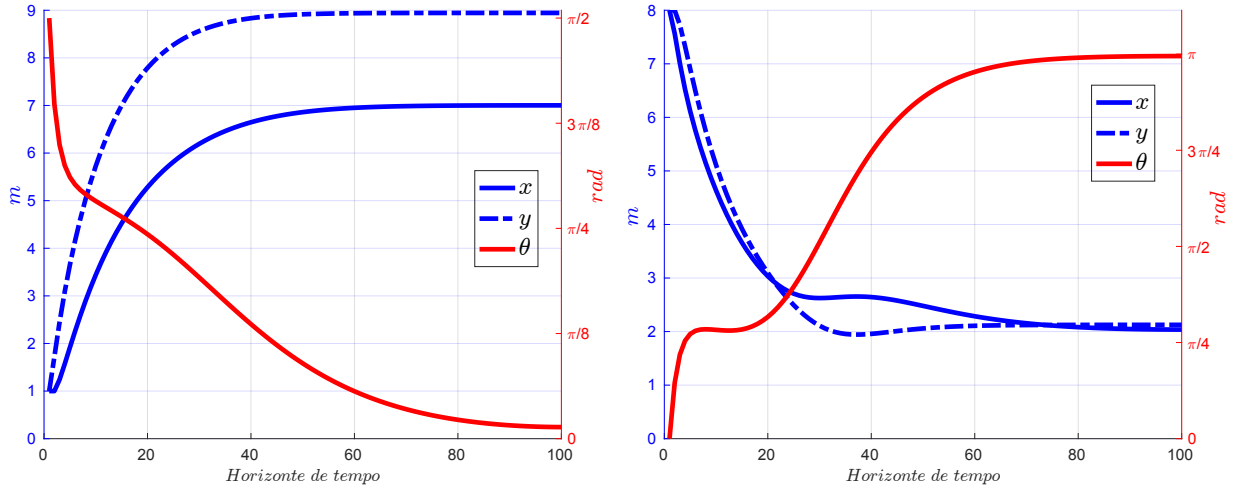
A variável x_{goal} é a configuração que se deseja alcançar. O termo $(x_t - x_{goal})^T L(x_t - x_{goal})$ codifica o custo de chegar até o alvo, e o termo $u_t^T W u_t$ é usado para penalizar grandes amplitudes de controle.

A seguir, mostramos os resultados de dois testes utilizando a PDD. Os parâmetros dos testes são exibidos na Tabela B.1, sendo x_0 a configuração inicial do robô e x_T a final. A sequência de controle aleatória U_0 é utilizada para gerar a trajetória inicial a partir de x_0 . Como o horizonte de tempo é de $100\Delta t$, a trajetória entre x_0 e x_T deve ser percorrida em 10 segundos (assumindo $\Delta t = 0.1s$). O primeiro teste convergiu para configuração desejada após 5 iterações e o segundo teste com 8 iterações.

Tabela B.1 – Parâmetros dos testes com a PDD (sem restrições).

Teste 1	Teste 2
$horizonte = 100\Delta t$	$horizonte = 100\Delta t$
$x_0 = (1, 1, \pi/2)^T$ e $x_T = (7, 9, 0)^T$	$x_0 = (8, 8, 0)^T$ e $x_T = (2, 2, \pi)^T$
$U_0 = rand(2, T - 1)$	$U_0 = rand(2, T - 1)$

A Figura B.1 exibe a evolução dos estados da trajetória ótima (segundo o funcional de custo das Equações (B.10) e (B.11)) que conecta x_0 e x_T para os dois testes.

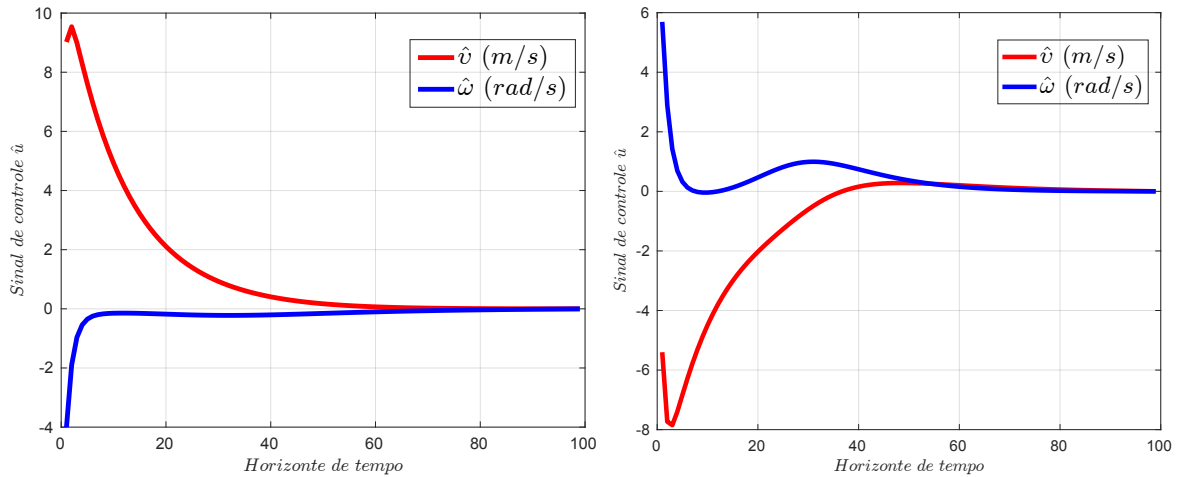


(a) Teste 1: $x_0 = (1, 1, \pi/2)^T$ e $x_T = (7, 9, 0)^T$.

(b) Teste 2: $x_0 = (8, 8, 0)^T$ e $x_T = (2, 2, \pi)^T$.

Figura B.1 – Evolução dos estados $(x, y, \theta)^T$ pertencentes às trajetórias ótimas.

Já a Figura B.2 mostra as saídas de controle que devem ser aplicadas para se atingir as trajetórias ótimas mostradas na Figura B.1. Como a velocidade linear \hat{v} da Figura B.2a é sempre maior que zero, podemos concluir que o robô andarรก sempre para frente no Teste 1. Contudo, na Figura B.2b observamos que a velocidade linear \hat{v} é menor que zero em grande parte do percurso do Teste 2, ou seja, para atingir a configuração desejada (sujeito ao funcional de custo estabelecido) o robô deverรก andar de ré.



(a) Teste 1.

(b) Teste 2.

Figura B.2 – Saídas de controle que devem ser aplicadas para se atingir as trajetórias ótimas.

Mostramos na Figura B.3 o custo ótimo da configuração inicial até a final. Como dito acima, a trajetória inicial é gerada randomicamente e está muito distante da trajetória ótima. Assim, os custos nas primeiras iterações são altos. Conforme as iterações evoluem, as novas trajetórias geradas se aproximam da ótima e o custo decai. Na Figura B.3, executamos a PDD até a 50ª iteração para comprovar que após a convergência para trajetória ótima o custo não se altera.

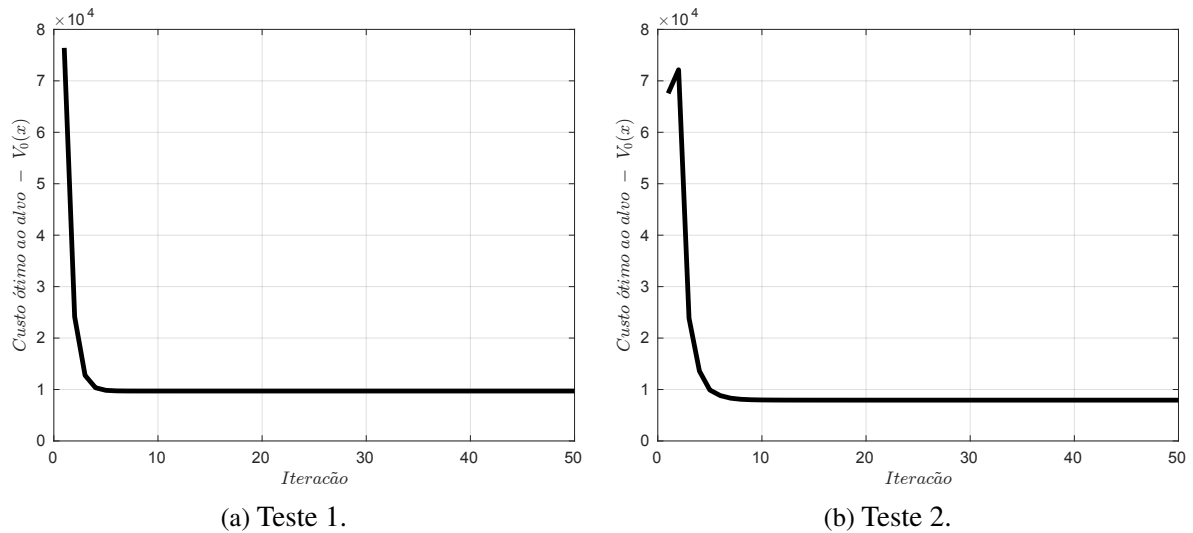


Figura B.3 – Custo ótimo para 50 iterações da PDD.

Pelas simulações acima, verificamos que a programação dinâmica diferencial pode ser utilizada como uma alternativa eficiente para gerar trajetórias ótimas para um robô não holonômico. As principais vantagens da PDD são sua robustez à trajetórias iniciais ruins (como as geradas randomicamente), e a sua aplicabilidade a uma grande escala de problemas. Por exemplo, no trabalho de Tassa et al. [2014] a PDD é usada para gerar trajetórias ótimas para um automóvel e para um humanoide, já no trabalho de Sun et al. [2018] a PDD é usada para gerar trajetórias para um drone. Nestas aplicações, modifica-se basicamente a dinâmica de controle $f(x_t, u_t)$ e atualiza-se as suas derivadas. Na próxima seção, vamos adicionar restrições de estados (incluir obstáculos) e de controle à estrutura da PDD.

B.3 Geração de trajetórias com restrições de controle e obstáculos

Estamos interessados em minimizar a expansão quadrática $Q(x, u)$ apresentada na Equação (B.3). A Equação (B.5) encontra a ação de controle δu^* que minimiza $Q(x, u)$. Entretanto, a Equação (B.5) não tem nenhuma restrição para obstáculos no ambiente nem restrições de velocidades. Em um problema real isto pode se tornar uma complicação, pois além de colidir com objetos presentes no ambiente, a PDD pode fornecer velocidades com amplitudes altas, implicando em movimentos bruscos do robô. Desta forma, manteremos a estrutura da PDD, mas incorporando as restrições de estados e de controle, de acordo com a Equação (B.12).

$$\begin{aligned} \min_{\delta u} \quad & Q(x, u) \\ \text{sujeito a} \quad & \underline{u} \leq u + \delta u \leq \bar{u} \\ & A' \delta u \leq b' \end{aligned} \quad (\text{B.12})$$

As velocidades \underline{u} e \bar{u} são as restrições de controle impostas. Se a estrutura da Equação (B.12) não fosse utilizada, poderíamos saturar a ação de controle \hat{u} fornecida pela Equação (B.8), ou seja, $\hat{u} = \min(\max(\hat{u}, \underline{u}), \bar{u})$. No entanto, a mudança na direção da ação de controle prejudicaria a convergência da PDD. Logo, uma alternativa plausível é introduzir a limitação de controle $\underline{u} \leq u + \delta u \leq \bar{u}$, enquanto se minimiza a expansão quadrática $Q(x, u)$.

Um obstáculo no ambiente pode ser representado por uma restrição de desigualdade $g(x) \leq 0$, isto é, se o robô estiver colidindo com o obstáculo teremos $g(x) \leq 0$, já se ele estiver em uma pose x livre de colisão teremos $g(x) > 0$. Assim, para que o robô evite um obstáculo podemos estabelecer a dinâmica (sendo κ uma constante positiva) [Kanoun et al., 2011]:

$$-\frac{dg(x)}{dt} \leq \kappa g(x). \quad (\text{B.13})$$

Para explicarmos a Equação (B.13), vamos multiplicá-la por -1, obtendo $dg/dt \geq -\kappa g$. O lado direito desta equação representa o quanto o robô poderá aproximar do obstáculo, ou seja, ter dg/dt negativo. Quanto menor for o valor de $g(x)$ ($g > 0$), mais perto o robô estará do obstáculo, e portanto menor é a taxa máxima de aproximação do obstáculo, $-\kappa g$. Por exemplo, se $g(x) = 1000$, teremos $dg/dt \geq -1000$ (supondo $\kappa = 1$). Isto significa que o robô pode se aproximar do obstáculo com qualquer velocidade ≤ 1000 . Porém, se $g(x) = 0.001$ obtemos $dg/dt \geq -0.001$, indicando que o robô só pode se aproximar do obstáculo, na melhor das hipóteses, com uma taxa muito pequena de 0.001. O que é natural, pois o robô está prestes a colidir com o obstáculo. Contudo, se $g(x) = 0$ teremos $dg/dt \geq 0$, e conseqüentemente, o robô só poderá manter a distância com o obstáculo $dg/dt = 0$ ou se afastar $dg/dt > 0$.

A variável κ determina o grau de afastamento do obstáculo. Quanto menor for o valor de κ , mais a trajetória penalizará o obstáculo. A função de distância $g(x)$ utilizada neste trabalho é apresentada na Equação (B.14) (embora não pareça, esta função é diferenciável). A variável ϕ receberá valor 1 se o estado x se encontrar fora do obstáculo e -1 se estiver dentro. A variável x_{ob} é o ponto do obstáculo mais perto de x .

$$g(x) = \|x - x_{ob}\| \phi. \quad (\text{B.14})$$

Durante a fase de testes, as restrições não lineares do robô não holonômico acabaram sendo rigorosas para a PDD gerar a trajetória, implicando em não convergência para a configuração desejada. Diante disto, decidimos considerar na PDD o modelo linear do robô holonômico¹² e utilizar as Equações (A.7) e (A.8) para o robô não holonômico seguir a trajetória de referência.

Desse modo, a aplicação da Equação (B.13) para o robô holonômico fornece:

¹² O robô holonômico pode se mover de forma independente nos eixos X e Y, e o seu modelo cinemático é:

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \Delta t.$$

$$-\frac{\partial g(x)}{\partial x} \frac{dx}{dt} \leq \kappa g(x), \quad (\text{B.15})$$

$$-\begin{bmatrix} (x-x_{ob})\phi & (y-y_{ob})\phi \end{bmatrix} \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix} \leq \kappa g(x). \quad (\text{B.16})$$

Como estamos interessados em ter uma equação em função de δu , substituímos \hat{v} por $\hat{v} - v$ e realizamos as modificações necessárias na Equação (B.16). Consequentemente, as componentes A' e b' da Equação (B.12) são obtidas (denotando $\tilde{x} = (x - x_{ob})$ e $\tilde{y} = (y - y_{ob})$):

$$\underbrace{-\begin{bmatrix} \tilde{x}\phi & \tilde{y}\phi \end{bmatrix}}_{A'} \underbrace{\begin{bmatrix} \hat{v}_x - v_x \\ \hat{v}_y - v_y \end{bmatrix}}_{\delta u} \leq \underbrace{\kappa g(x) + \begin{bmatrix} \tilde{x}\phi & \tilde{y}\phi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}}_{b'}. \quad (\text{B.17})$$

Teremos uma restrição $A' \delta u \leq b'$ para cada obstáculo presente no ambiente. Na minimização da Equação (B.12) os termos de $Q(x, u)$ que não dependem de δu serão irrelevantes, então podemos reescrever a Equação (B.12):

$$\begin{aligned} \min_{\delta u} \quad & \delta u^T Q_{uu} \delta u + [Q_u + \delta x^T Q_{ux}] \delta u \\ \text{sujeito a} \quad & \underline{u} - u \leq \delta u \leq \bar{u} - u \\ & A' \delta u \leq b' \end{aligned} \quad (\text{B.18})$$

Portanto, na implementação da PDD com restrições usaremos a Equação (B.18) para determinar δu^* . Nos testes que mostraremos nesta seção, utilizamos a função *quadprog* do Matlab para resolver a minimização da Equação (B.18). Também consideramos que os obstáculos têm a forma de um polígono, e criamos uma rotina para identificar o ponto x_{ob} mais próximo de x . Nos testes, utilizamos uma sequência de controle inicial aleatória e $\Delta t = 0.05$. Em todos eles a PDD convergiu para configuração desejada na primeira iteração.

No primeiro teste ilustrado na Figura B.4, verificamos a influência do parâmetro κ na trajetória e comprovamos que quanto menor for o valor deste parâmetro, mais a trajetória desviará do obstáculo. Neste teste limitamos as velocidades em $-1\text{m/s} \leq \hat{v}_x, \hat{v}_y \leq 1\text{m/s}$, e podemos verificar na Figura B.5 que esta restrição foi atendida para um horizonte de $700 \Delta t$.

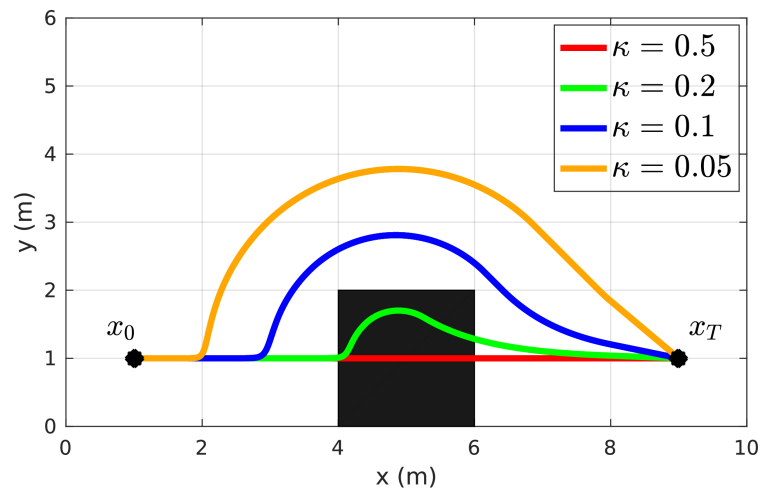
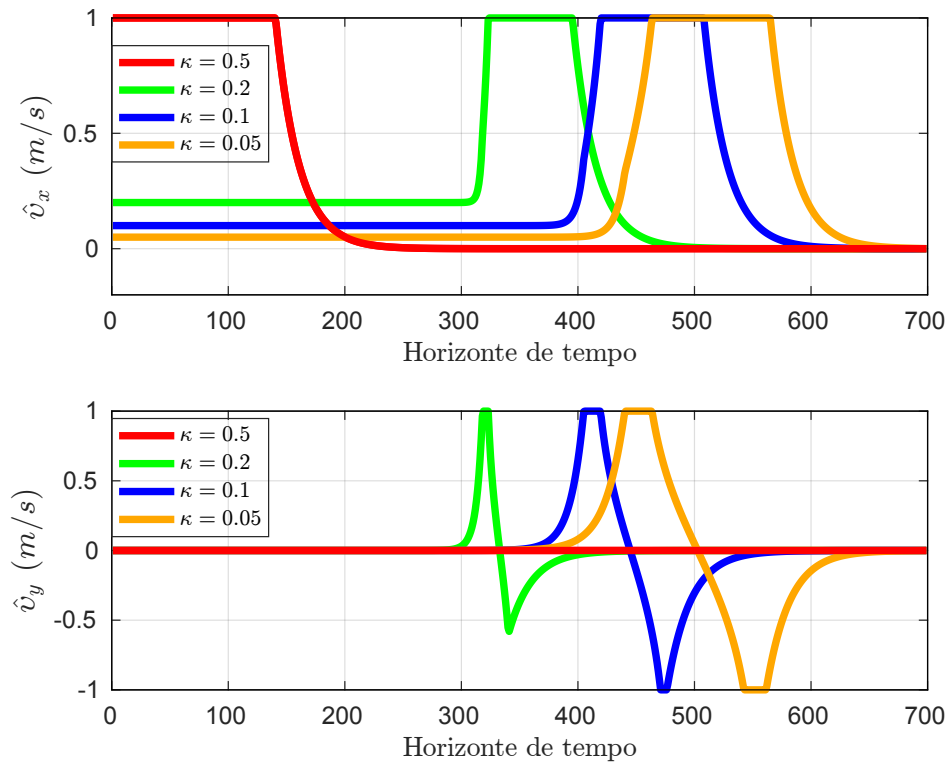
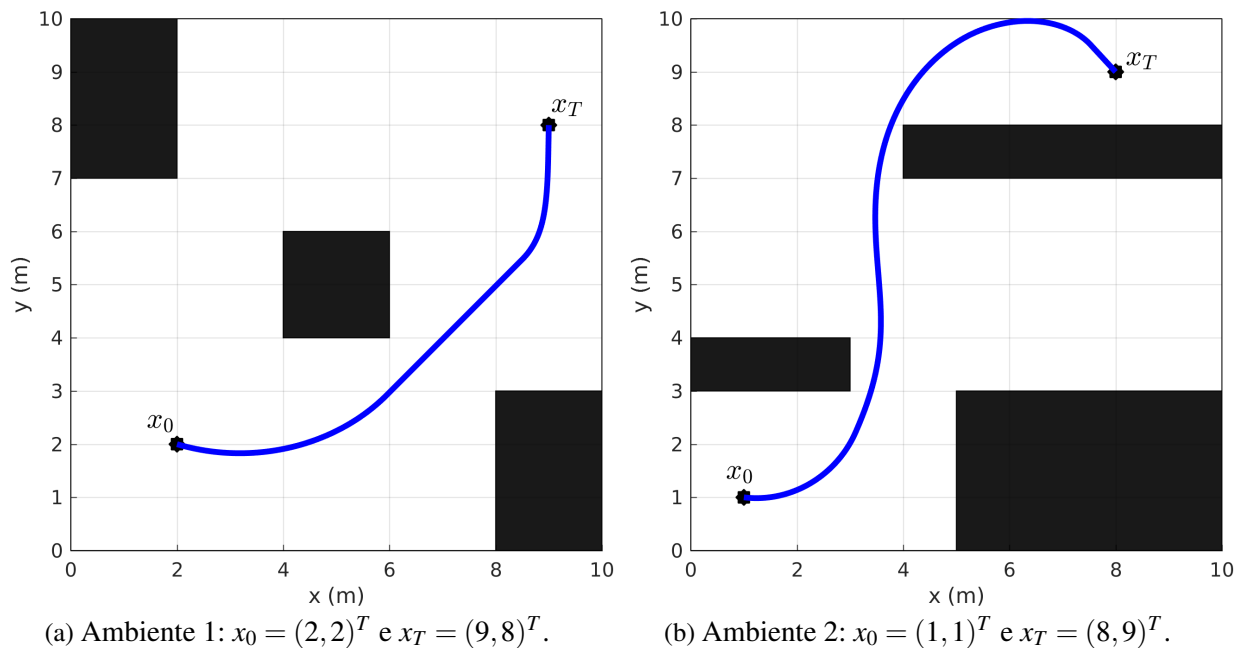


Figura B.4 – Influência do parâmetro κ na penalização de um obstáculo, utilizando $x_0 = (1, 1)^T$ e $x_T = (9, 1)^T$.


 Figura B.5 – Sinal de controle \hat{u} para o teste apresentado na Figura B.4.

Na Figura B.6 temos dois testes em ambientes distintos, possuindo três obstáculos cada um. No Ambiente 1 usamos a restrição de controle $-0.5m/s \leq \hat{v}_x, \hat{v}_y \leq 0.5m/s$ e no Ambiente 2 a restrição $-1.5m/s \leq \hat{v}_x, \hat{v}_y \leq 1.5m/s$. Em ambos, o horizonte de tempo foi de $500 \Delta t$ e $\kappa = 0.1$. No Ambiente 1, os obstáculos são mais simples e a trajetória até a configuração desejada é mais suave, em comparação com a trajetória gerada no Ambiente 2.


 (a) Ambiente 1: $x_0 = (2, 2)^T$ e $x_T = (9, 8)^T$.

 (b) Ambiente 2: $x_0 = (1, 1)^T$ e $x_T = (8, 9)^T$.

 Figura B.6 – Evolução dos estados $(x, y)^T$ pertencentes à trajetória ótima em ambientes com obstáculos.

As saídas de controle que devem ser aplicadas, para se atingir as trajetórias ótimas mostradas na Figura B.6, são exibidas na Figura B.7. Vídeos ilustrando um robô não holonômico seguindo as trajetórias geradas pela PDD podem ser visualizados em <https://youtu.be/6jlcMzEkjs4> (Ambiente 1) e em <https://youtu.be/Ai4AEQYizwE> (Ambiente 2).

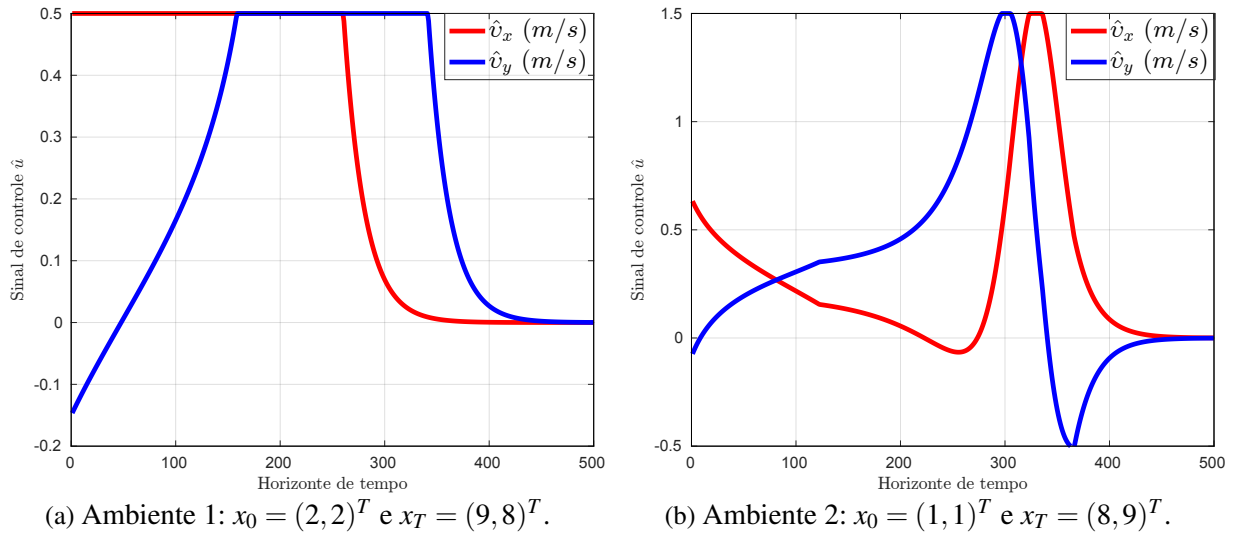


Figura B.7 – Saídas de controle que devem ser aplicadas para se atingir as trajetórias ótimas.

Por meio destas simulações, podemos concluir que a PDD pode ser utilizada para gerar uma trajetória ótima entre duas configurações, sob restrições de controle e de obstáculos, e poderá ser utilizada em conjunto com o PDMPO apresentado no Capítulo 4. Nesta aplicação, a função da PDD seria gerar uma trajetória factível de ser seguida pelo robô quando o PDMPO determinar uma nova política de mudança de estados.

REFERÊNCIAS

- Aberdeen, D. (2003). A survey of approximate methods for solving partially observable markov decision process. *Report National ICT Australia*. Citado na página 72.
- Abreu, A. (2014). Robot localization from minimalist inertial data using a hidden markov model. In *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, pages 247–252. IEEE. Citado na página 37.
- Al-Wazzan, A., Al-Farhan, R., Al-Ali, F., and El-Abd, M. (2016). Tour-guide robot. In *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*, pages 1–5. IEEE. Citado na página 32.
- Antonelli, G., Caccavale, F., Grossi, F., and Marino, A. (2010). Simultaneous calibration of odometry and camera for a differential drive mobile robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 5417–5422. IEEE. Citado na página 49.
- Bacalhau, E. T. et al. (2015). *Otimização de políticas de manutenção em redes de distribuição de energia elétrica por estratégias híbridas baseadas em programação dinâmica*. Tese de doutorado, Universidade Estadual de Campinas, Campinas, SP. Citado 3 vezes nas páginas: 89, 90 e 91.
- Baker, J. (1975). The dragon system—an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29. Citado na página 38.
- Balbinot, A. and Brusamarello, V. (2010). Instrumentação e fundamentos de medidas—vol. 1, vol. 1. *Rio de Janeiro, Brazil: LTC*. Citado 2 vezes nas páginas: 25 e 26.
- Ballard, D. H. and Brown, C. M. (1982). Computer vision. *J: Prentice Hall*. Citado 2 vezes nas páginas: 26 e 27.
- Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363. Citado na página 37.
- Baum, L. E. and Sell, G. R. (1968). Growth functions for transformations on manifolds. *Pac. J. Math.*, 27(2):211–227. Citado 2 vezes nas páginas: 37 e 38.
- Baumeister, J. and Leitão, A. (2014). *Introdução à teoria de controle e programação dinâmica*, volume 1. IMPA. Citado na página 90.

- Bellman, R. (1957a). Dynamic programming. *Princeton University Press*. Citado na página 90.
- Bellman, R. (1957b). A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684. Citado na página 66.
- Bengio, Y. and Frasconi, P. (1996). Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249. Citado na página 45.
- Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA. Citado na página 90.
- Bhowmik, A. K. (2017). Sensification of computing: adding natural sensing and perception capabilities to machines. *APSIPA Transactions on Signal and Information Processing*, 6. Citado na página 28.
- Bonin-Font, F., Ortiz, A., and Oliver, G. (2008). Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296. Citado na página 26.
- Braziunas, D. (2003). Pomdp solution methods. *University of Toronto*. Citado na página 72.
- Briers, M., Doucet, A., and Maskell, S. (2010). Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61. Citado na página 47.
- Calinon, S. (2017). Hidden markov models and extensions. Disponível em: <<http://calinon.ch/misc/EE613/EE613-slides-7-HMM.pdf>>. Acesso em: 21 de dez. de 2019. Citado na página 38.
- Candy, J. V. (2016). *Bayesian signal processing: classical, modern, and particle filtering methods*, volume 54. John Wiley & Sons. Citado na página 38.
- Cappé, O., Moulines, E., and Rydén, T. (2009). Inference in hidden markov models. In *Proceedings of EUSFLAT conference*, pages 14–16. Citado na página 38.
- Cardoso, R. T. N. (2008). *Ferramentas para programação dinâmica em malha aberta*. Tese de doutorado, Universidade Federal de Minas Gerais, Belo Horizonte, MG. Citado na página 90.
- Cassandras, C. G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science & Business Media. Citado 2 vezes nas páginas: 33 e 34.
- Charles, R., Bry, A., and Roy, N. (2013). Polynomial trajectory planning for quadrotor flight. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*. Citado na página 89.
- Chernov, V., Alander, J., and Bochko, V. (2015). Integer-based accurate conversion between rgb and hsv color spaces. *Computers & Electrical Engineering*, 46:328–337. Citado na página 27.

- Chong, K. S. and Kleeman, L. (1997). Accurate odometry and error modelling for a mobile robot. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2783–2788. IEEE. Citado na página 25.
- Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press. Citado na página 23.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press. Citado na página 23.
- Correa, D. S. O., Sciotti, D. F., Prado, M. G., Sales, D. O., Wolf, D. F., and Osorio, F. S. (2012). Mobile robots navigation in indoor environments using kinect sensor. In *2012 Second Brazilian Conference on Critical Embedded Systems*, pages 36–41. IEEE. Citado na página 32.
- Corso, N. (2013). Loop closure transformation estimation and verification using 2d lidar scanners. *Technical Report No. UCB/EECS-2013-73*. Citado na página 31.
- Duarte, A. F. M. (2015). *Reconhecimento de objetos baseado em visão artificial*. Dissertação de mestrado, Universidade do Porto, Porto. Citado na página 27.
- Dudek, G. and Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge university press. Citado 7 vezes nas páginas: 17, 22, 24, 25, 26, 29 e 30.
- Dymarski, P. (2011). *Hidden Markov Models: Theory and Applications*. BoD–Books on Demand. Citado na página 37.
- Elliott, R. J., Aggoun, L., and Moore, J. B. (2008). *Hidden Markov models: estimation and control*, volume 29. Springer Science & Business Media. Citado na página 38.
- Feinberg, E. A. and Shwartz, A. (2012). *Handbook of Markov decision processes: methods and applications*, volume 40. Springer Science & Business Media. Citado 2 vezes nas páginas: 66 e 67.
- Flir (2019). Visão estéreo precisa e flexível. Disponível em: <<https://prod.flir.com.br/iis/machine-vision/stereo-vision/>>. Acesso em: 20 de nov. de 2019. Citado na página 28.
- Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278. Citado na página 38.
- Françoise, J., Temps-Réel, I., Caramiaux, B., and Bevilacqua, F. (2011). *Realtime Segmentation and Recognition of Gestures using Hierarchical Markov Models*. PhD thesis, Master’s Thesis, Université Pierre et Marie Curie, Ircam. Citado 2 vezes nas páginas: 44 e 45.

- Fraser, A. M. (2008). *Hidden Markov models and dynamical systems*, volume 107. Siam. Citado na página 47.
- Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292. Citado na página 52.
- Ghosh, A. (2016). Slam (robotics). Disponível em: <<https://arunabh98.github.io/summer%20project/2016/05/21/slam/>>. Acesso em: 10 de nov. de 2019. Citado na página 25.
- Gomez, C., Hernandez, A. C., Crespo, J., and Barber, R. (2016). A topological navigation system for indoor environments based on perception events. *International Journal of Advanced Robotic Systems*, 14(1):1729881416678134. Citado na página 37.
- Gomez, C., Hernandez, A. C., Crespo, J., and Barber, R. (2017). Uncertainty-based localization in a topological robot navigation system. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 67–72. Citado 2 vezes nas páginas: 33 e 37.
- Howard, R. A. (1960). *Dynamic programming and markov processes*. John Wiley & Sons. Citado na página 66.
- Hu, Q. and Yue, W. (2007). *Markov decision processes with their applications*, volume 14. Springer Science & Business Media. Citado na página 66.
- Jacobson, D. H. and Mayne, D. Q. (1970). *Differential dynamic programming*. North-Holland. Citado 2 vezes nas páginas: 90 e 91.
- Jamali, A., Rahman, A. A., Boguslawski, P., Kumar, P., and Gold, C. M. (2017). An automated 3d modeling of topological indoor navigation network. *GeoJournal*, 82(1):157–170. Citado na página 32.
- Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134. Citado na página 70.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45. Citado na página 46.
- Kanoun, O., Lamiroux, F., and Wieber, P.-B. (2011). Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, 27(4):785–792. Citado na página 95.
- Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894. Citado na página 24.

- Krishnamurthy, V. (2016). *Partially observed Markov decision processes*. Cambridge University Press. Citado na página 71.
- Litomisky, K. (2012). Consumer rgb-d cameras and their applications. *Rapport technique, University of California*, 20. Citado na página 28.
- Liu, F., Wang, J., Zhang, J., and Han, H. (2019). An indoor localization method for pedestrians base on combined uwb/pdr/floor map. *Sensors*, 19(11):2578. Citado na página 32.
- Llarena, A., Savage, J., Kuri, A., and Escalante-Ramírez, B. (2012). Odometry-based viterbi localization with artificial neural networks and laser range finders for mobile robots. *Journal of Intelligent & Robotic Systems*, 66(1-2):75–109. Citado na página 60.
- Lomasney, E. (2019). Moving from ccd to cmos cameras ccd sensors go obsolete. Disponível em: <<https://www.adimec.com/moving-from-ccd-towards-cmos-cameras-as-many-ccd-sensors-go-obsolete/>>. Acesso em: 21 de nov. de 2019. Citado 2 vezes nas páginas: 26 e 27.
- Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548. Citado na página 72.
- MathWorks, T. (2005). Color approximation. Disponível em: <<http://matlab.izmiran.ru/help/toolbox/images/color4.html>>. Acesso em: 20 de nov. de 2019. Citado na página 27.
- Microsoft (2019). Acessórios: Kinect. Disponível em: <<https://support.xbox.com/pt-BR/browse/xbox-360/accessories/Kinect>>. Acesso em: 21 de nov. de 2019. Citado na página 28.
- Monteiro, N. S., Maia, C. A., and Gonçalves, V. M. (2019a). 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, Ouro Preto*. Citado na página 20.
- Monteiro, N. S., Maia, C. A., and Gonçalves, V. M. (2019b). Estimação da localização de um robô não holonômico utilizando modelo oculto de markov (hmm) e o filtro de kalman estendido (ekf). In: *Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI), 2019, Ouro Preto*. Citado na página 20.
- Monteiro, N. S., Maia, C. A., and Gonçalves, V. M. (2019c). Geração de trajetórias Ótimas para o robô não holonômico usando programação dinâmica diferencial. In: *Anais da 14ª Conferência Brasileira de Dinâmica Controle e Aplicações (DINCON), 2019, São Carlos*. Citado na página 21.

- Montero, A. S., Sekkati, H., Lang, J., Laganière, R., and James, J. (2015). Framework for natural landmark-based robot localization. Disponível em: <<http://www.solism.ca/projects/planar.html>>. Acesso em: 8 de nov. de 2019. Citado na página 25.
- Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262. Citado na página 51.
- Murphy, K. P. (2000a). A survey of pomdp solution techniques. *environment*, 2:X3. Citado na página 72.
- Murphy, R. R. (2000b). *Introduction to AI Robotics*. MIT Press. Citado 7 vezes nas páginas: 17, 25, 26, 27, 28, 29 e 30.
- Muslim, M. A. and Ishikawa, M. (2008). Formation of graph-based maps for mobile robots using hidden markov models. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)*. *IEEE International Joint Conference on*, pages 3099–3105. IEEE. Citado na página 37.
- Nardi, L. and Stachniss, C. (2019). Uncertainty-aware path planning for navigation on road networks using augmented mdps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. Citado na página 66.
- Nashed, S. B., Ilstrup, D. M., and Biswas, J. (2018). Localization under topological uncertainty for lane identification of autonomous vehicles. *arXiv preprint arXiv:1803.01378*. Citado na página 37.
- Nieuwenhuisen, M., Droeschel, D., Holz, D., Stückler, J., Berner, A., Li, J., Klein, R., and Behnke, S. (2013). Mobile bin picking with an anthropomorphic service robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 2327–2334. IEEE. Citado na página 32.
- Páll, E., Tamás, L., and Buşoniu, L. (2016). Analysis and a home assistance application of online aems2 planning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5013–5019. IEEE. Citado 2 vezes nas páginas: 65 e 66.
- Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450. Citado na página 72.
- Pellegrini, J. and Wainer, J. (2007). Processos de decisão de markov: um tutorial. *Revista de Informática Teórica e Aplicada*, 14(2):133–179. Citado 7 vezes nas páginas: 65, 67, 68, 69, 70, 71 e 74.
- Peter, O. (2016). *Learning Complex Markov Models for Mobile Robotics*. PhD thesis, University of Oxford, England. Citado 3 vezes nas páginas: 33, 34 e 35.

- Pineau, J., Gordon, G., and Thrun, S. (2004). *Tractable planning under uncertainty: exploiting structure*. PhD thesis, Carnegie Mellon University, the Robotics Institute. Citado na página 65.
- Puterman, M. L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons. Citado 2 vezes nas páginas: 67 e 69.
- Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Elsevier. Citado 6 vezes nas páginas: 34, 35, 37, 38, 39 e 41.
- Rahman, S. and Zelinsky, A. (1999). Mobile robot navigation based on localisation using hidden markov models. In *Australasian Conference on Robotics and Automation, ACRA*. Citado na página 36.
- Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,. Citado na página 47.
- Särkkä, S. (2013). *Bayesian filtering and smoothing*, volume 3. Cambridge University Press. Citado na página 48.
- Savage, J., Fuentes, O., Contreras, L., and Negrete, M. (2018). Map representation using hidden markov models for mobile robot localization. In *MATEC Web of Conferences*, volume 161, page 03011. EDP Sciences. Citado na página 37.
- Song, G., Wang, H., Zhang, J., and Meng, T. (2011). Automatic docking system for recharging home surveillance robots. *IEEE Transactions on Consumer Electronics*, 57(2):428–435. Citado na página 32.
- Source, T. I. (2019). 3d stereo camera system. Disponível em: <https://www.theimagingsource.com/newsletter-2.0/20171004/body.en_US.phtml>. Acesso em: 25 de nov. de 2019. Citado na página 28.
- Souto, R. P. (2003). Segmentação de imagem multiespectral utilizando-se o atributo matiz. *Instituto Nacional de Pesquisas Espaciais-INPE*. Citado na página 27.
- Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control*, volume 3. Wiley New York. Citado 2 vezes nas páginas: 52 e 88.
- Sucar, L. E. (2011). *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions: Concepts and Solutions*. IGI Global. Citado 3 vezes nas páginas: 67, 70 e 71.
- Sun, W., Pan, Y., Lim, J., Theodorou, E. A., and Tsiotras, P. (2018). Min-max differential dynamic programming: Continuous and discrete time formulations. *Journal of Guidance, Control, and Dynamics*, 41(12):2568–2580. Citado na página 94.

- Tassa, Y., Mansard, N., and Todorov, E. (2014). Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE. Citado na página 94.
- Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., et al. (1999). Minerva: A second-generation museum tour-guide robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 3. IEEE. Citado na página 32.
- Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press. Citado 20 vezes nas páginas: 18, 22, 24, 25, 29, 30, 31, 35, 38, 46, 47, 48, 65, 67, 68, 69, 70, 71, 72 e 73.
- Veloso, M. M., Biswas, J., Coltin, B., and Rosenthal, S. (2015). Cobots: Robust symbiotic autonomous mobile service robots. In *IJCAI*, page 4423. Citado na página 32.
- Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269. Citado na página 38.
- Wang, C., Cheng, J., Wang, J., Li, X., and Meng, M. Q.-H. (2018). Efficient object search with belief road map using mobile robot. *IEEE Robotics and Automation Letters*, 3(4):3081–3088. Citado na página 66.
- Yakoubi, M. A. and Laskri, M. T. (2016). The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of innovation in digital ecosystems*, 3(1):37–43. Citado na página 32.
- Yin, J. and Meng, Y. (2009). Abnormal behavior recognition using self-adaptive hidden markov models. In *International Conference Image Analysis and Recognition*, pages 337–346. Springer. Citado na página 36.
- Zhu, J., Li, Q., Cao, R., Sun, K., Liu, T., Garibaldi, J., Li, Q., Liu, B., and Qiu, G. (2019). Indoor topological localization using a visual landmark sequence. *Remote Sensing*, 11(1):73. Citado na página 37.