

Danilo Giacomin Schneider

**SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO 3D
SIMULTÂNEOS EM ROBÔS MÓVEIS:
FUSÃO DE SENSORES PARA LOCALIZAÇÃO COM
FILTRO DE KALMAN ESTENDIDO**

Dissertação submetida ao Programa
de Pós-Graduação em Engenharia de
Automação e Sistemas para a obten-
ção do Grau de Mestre em Engenharia
de Automação e Sistemas.
Orientador
Universidade Federal de Santa Cata-
rina: Prof. Dr.-Ing. Marcelo Ricardo
Stemmer

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Schneider, Danilo Giacomin

Sistema de localização e mapeamento 3D simultâneos
em robôs móveis : fusão de sensores para localização
com filtro de Kalman estendido / Danilo Giacomin
Schneider ; orientador, Marcelo Ricardo Stemmer,
2018.

113 p.

Dissertação (mestrado) - Universidade Federal de
Santa Catarina, Centro Tecnológico, Programa de Pós
Graduação em Engenharia de Automação e Sistemas,
Florianópolis, 2018.

Inclui referências.

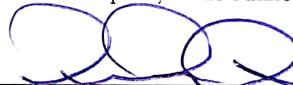
1. Engenharia de Automação e Sistemas. 2. Robótica
Móvel. 3. RGB-D SLAM. 4. Fusão de sensores. 5.
Filtro de Kalman Estendido. I. Stemmer, Marcelo
Ricardo. II. Universidade Federal de Santa
Catarina. Programa de Pós-Graduação em Engenharia de
Automação e Sistemas. III. Título.

Danilo Giacomin Schneider

**SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO 3D
SIMULTÂNEOS EM ROBÔS MÓVEIS: FUSÃO DE
SENSORES PARA LOCALIZAÇÃO COM FILTRO DE
KALMAN ESTENDIDO**

Esta Dissertação foi julgada aprovada para a obtenção do Título de “Mestre em Engenharia de Automação e Sistemas”, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 14 de Junho 2018.



Prof. Dr. Werner Kraus Junior

Coordenador

Universidade Federal de Santa Catarina

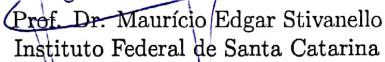
Banca Examinadora:



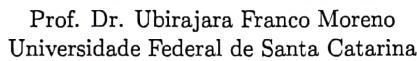
Prof. Dr.-Ing. Marcelo Ricardo Stemmer

Orientador

Universidade Federal de Santa Catarina



Prof. Dr. Maurício Edgar Stivanello
Instituto Federal de Santa Catarina



Prof. Dr. Ubirajara Franco Moreno
Universidade Federal de Santa Catarina



Prof. Dr. Tiago Loureiro Figaro da Costa Pinto
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus queridos pais.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina (UFSC) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelos recursos e conhecimentos proporcionados. Aos meus familiares e amigos próximos, sobretudo aos meus pais, pelo incentivo. A todos os meus professores, em especial ao orientador, Prof. Dr. -Ing. Marcelo Ricardo Stemmer, pela minha formação.

O aspecto mais triste da vida atual é que a ciência ganha em conhecimento mais rapidamente do que a sociedade em sabedoria.

(Isaac Asimov, 1988)

RESUMO

Para que um robô seja inteiramente autônomo ele precisa de alguma forma de representação do ambiente no qual ele está presente, assim como a sua localização em relação a este ambiente. Localizar-se com precisão e construir um mapa do ambiente permite que o robô possa alcançar metas pré-determinadas. A localização é uma das tarefas mais importantes para a navegação autônoma de robôs móveis, onde robustez e precisão são características fundamentais. Essa tarefa baseia-se no mapa do seu ambiente. Em contrapartida, a construção do mapa do ambiente depende da localização precisa do robô móvel. Em ambientes desconhecidos este é um processo conflitante, mas correlacionado. A introdução de sistemas de visão computacional em robôs móveis significa um aumento expressivo em suas habilidades sensoriais, o que implica em uma maior versatilidade e segurança nas aplicações do robô. Técnicas probabilísticas podem ser usadas em robótica para correção dos eventuais erros dos sensores, assim como técnicas de fusão temporal e fusão multissensorial. Este trabalho apresenta a implementação, através do ROS, de um Filtro de Kalman Estendido para a fusão de três fontes de sensoriamento diferentes e estimar a posição e orientação do robô de forma mais robusta, precisa e, consequentemente, realizar um mapeamento também mais fiel do ambiente em 3D, com registro na forma de nuvem de pontos, obtida através das imagens de cor e profundidade de uma câmera Kinect acoplada ao robô.

Palavras-chave: Filtro de Kalman Estendido. EKF. Robô Móvel. RGB-D SLAM. SLAM 3D. Kinect. ROS.

ABSTRACT

In order to be entirely autonomous a robot needs some sort of environment representation in which it is present as well as its localization in this environment. Finding itself accurately and building a map of the environment allows the robot to reach predetermined goals. Localization is one of the most important tasks for autonomous navigation of mobile robots, where robustness and precision are fundamental characteristics. This task is based on the map of the environment. In contrast, building this map depends on the precise location of the mobile robot. In unknown environments this is a conflicting but correlated process. The introduction of computer vision systems in mobile robots means an expressive increase in their sensorial abilities, which implies in a greater versatility and security in the robot's applications. Probabilistic techniques can be used in robotics to correct eventual sensor noises, as well as temporal fusion and multisensory fusion techniques. The present work describes the implementation of an Extended Kalman Filter using ROS to perform the fusion of three different sensory sources and estimate the position and orientation of the robot more accurately and robustly and, consequently, resulting in a more accurate mapping of the 3D environment, with point cloud registry obtained through the color and depth images of a Kinect camera coupled to the robot.

Keywords: Extended Kalman Filter. EKF. Mobile Robot. RGB-D SLAM. SLAM 3D. Kinect. ROS.

LISTA DE FIGURAS

Figura 1 Autorretrato do robô Curiosity da NASA em Marte.....	32
Figura 2 O quadro de referência global (X_I, Y_I) e o quadro de referência (X_R, Y_R) local do robô em 2D.....	33
Figura 3 Propagação de erro na posição de um robô móvel diferencial.....	38
Figura 4 Ilustração de um exemplo EKF-SLAM.....	48
Figura 5 Exemplo de construção de grafo. Restrições entre nós no grafo são representadas com restrições suaves (como molas).....	49
Figura 6 Descriptor SIFT	54
Figura 7 O padrão de amostragem BRISK com $N = 60$ pontos: os pequenos círculos azuis indicam os locais de amostragem; os maiores, círculos vermelhos tracejados, são desenhados num raio σ correspondente ao desvio padrão do núcleo Gaussiano utilizado para suavizar os valores de intensidade nos pontos de amostragem.	60
Figura 8 Exemplo de combinação com BRISK. As combinações resultantes são conectadas por linhas verdes, que não mostram falsos positivos claros.....	61
Figura 9 (a) Densidade de células ganglionares sobre a retina (b) Áreas da retina.....	62
Figura 10 Ilustração do padrão de amostragem FREAK em que cada círculo representa um campo receptivo onde a imagem é suavizada com o respectivo núcleo Gaussiano.....	63
Figura 11 Aplicação simples do algoritmo RANSAC em um conjunto bidimensional de dados.....	65
Figura 12 Etapas de um processo de registro (neste caso, cantos foram utilizados como características)	68
Figura 13 Nuvem de pontos obtida pelo sensor Kinect com auxílio da biblioteca PCL.....	69
Figura 14 Saída do <i>ekf_localization_node</i> (verde) na fusão de dados de odometria, duas IMUs e duas unidades GPS. A trajetória do robô como uma media dos dados brutos dos GPS é mostrada em vermelho.....	72
Figura 15 Foto do robô Pioneer 3-DX montado com os sensores usados no projeto.....	81
Figura 16 Imagem dos componentes internos do Kinect.....	84

Figura 17 Diagrama simplificado do sistema de localização e mapeamento remotos usado no projeto.....	87
Figura 18 Gráfico da posição (x,y) e orientação (γ) absolutas reportados pelos sensores: Odometria encoders (vermelho), Odometria visual (azul), pose do laser (verde). E o resultado da fusão pelo EKF (preto).....	91
Figura 19 Gráfico das velocidade linear (\dot{x}) e angular ($\dot{\gamma}$) relativas reportados pelos sensores: Odometria encoders (vermelho), Odometria visual (azul). E o resultado da fusão pelo EKF (preto).....	92
Figura 20 Gráfico da trajetória (x,y) do robô vista de cima. As legendas se encontram na figura.	92
Figura 21 Gráfico das covariâncias reportadas pelos sensores e da fusão pelo EKF pelo tempo. As legendas se encontram na figura..	93
Figura 22 Gráfico com falha na odometria visual. As legendas se encontram na figura.	94
Figura 23 Gráfico do erro translacional obtido pelo algoritmo <i>evaluate_rpe.py</i>	95
Figura 24 Comparação de trajetórias obtida pelo algoritmo <i>evaluate_ate.py</i> . As legendas se encontram na imagem.....	96
Figura 25 Grades de ocupação geradas pelo pacote <i>rtabmap_ros</i> nos casos: a) Localização EKF; b) Odometria visual.	96
Figura 26 Visualização em perspectiva do mapa 3D de nuvem de pontos gerado pelo pacote <i>rtabmap_ros</i> (trajetória do robô em azul). 97	97
Figura 27 Foto em perspectiva do ambiente capturada por uma câmera convencional.....	98
Figura 28 Gráfico da trajetória (x,y) do robô vista de cima com a pose via laser reportando valores errados. As legendas se encontram na figura.	99
Figura 29 Resposta do sistema com a configuração alternativa....	99
Figura 30 Árvore de <i>frames</i>	113

LISTA DE TABELAS

Tabela 1	Valores padrões dos parâmetros do P3-DX.....	82
Tabela 2	Performance e características do LMS200.....	83
Tabela 3	Especificações do Microsoft Kinect v1.....	84
Tabela 4	Tabela com as variáveis do espaço de estados que são medidas pelos sensores e as utilizadas para fusão pelo EKF.....	85
Tabela 5	<i>Relative pose error</i> (RPE) [m]	94
Tabela 6	<i>Absolute trajectory error</i> (ATE) [m]	95
Tabela 7	Tabela com a configuração alternativa das variáveis do espaço de estados que são utilizadas para fusão pelo EKF.....	97

LISTA DE ABREVIATURAS E SIGLAS

AMR	<i>Autonomous Mobile Robot</i>
SLAM	<i>Simultaneous Localization and Mapping</i>
IMU	<i>Inertial Measurement Unit</i>
GPS	<i>Global Positioning System</i>
LRF	<i>Laser Rangefinder</i>
SfM	<i>Structure from Motion</i>
ROS	<i>Robot Operating System</i>
EKF	<i>Extended Kalman Filter</i>
UKF	<i>Unscented Kalman Filter</i>
GRV	<i>Gaussian Random Variable</i>
SIFT	<i>Scale-invariant Feature Transform</i>
DeG	Diferença-entre-Gaussianas
SURF	<i>Speed-Up Robust Features</i>
LoG	Laplaciano do Gaussiano
BRIEF	<i>Binary Robust Independent Elementary Features</i>
LSH	<i>Local Sensitive Hashing</i>
PCA	<i>Principal Component Analysis</i>
LDA	<i>Linear Discriminant Analysis</i>
ORB	<i>Oriented FAST and Rotated BRIEF</i>
FAST	<i>Features from Accelerated Segment Test</i>
BRISK	<i>Binary Robust Invariant Scalable Keypoints</i>
AGAST	<i>Adaptive and Generic Accelerated Segment Test</i>
FREAK	<i>Fast Retina Keypoint</i>
RANSAC	<i>Random Sample Consensus</i>
MSSs	<i>Minimum Sample Sets</i>
CS	<i>Consensus Set</i>
ICP	<i>Iterative Closest Point</i>
CPs	<i>Control Points</i>
PCL	<i>Point Cloud Library</i>
LiDAR	<i>Light Detection And Ranging</i>
Dof	<i>Degrees-of-Freedom</i>
MAP	<i>Maximum a Posteriori</i>

LIOP	<i>Local Intensity Order Pattern</i>
MRRID	<i>Multisupport Region Rotation and Intensity Monotonic Invariant Descriptor</i>
MROGH	<i>MultiSupport Region Order-Based Gradient Histogram</i>
RPE	<i>Relative Pose Error</i>
ATE	<i>Absolute Trajectory Error</i>
PID	<i>Proportional-Integral-Derivative</i>
PWM	<i>Pulse Width Modulation</i>
FPS	<i>Frames-Per-Second</i>
GFTT	<i>Good Features To Track</i>
RMSE	<i>Root Mean Square Error</i>

LISTA DE SÍMBOLOS

x	Translação em relação ao eixo x.
y	Translação em relação ao eixo y.
z	Translação em relação ao eixo z.
α	Rotação em relação ao eixo x. Ângulo de rolagem.
β	Rotação em relação ao eixo y. Ângulo de arfagem.
γ	Rotação em relação ao eixo z. Ângulo de guinada.
\dot{x}	Velocidade linear relativa.
$\dot{\gamma}$	Velocidade angular relativa em torno do eixo z.
s_k	Vetor de estados do sistema, no instante k .
w_{k-1}	Ruído de processo.
v_k	Ruído de medição.
H	Matriz de observação.
P_k	Matriz de covariâncias do erro no instante k .
Q	Matriz de covariâncias do ruído de processo.
R	Matriz de covariâncias do ruído de medição.
K	Ganho de Kalman.
\int	Integral
Σ	Somatório
Δ	Delta (variação)

SUMÁRIO

1	INTRODUÇÃO	25
1.1	OBJETIVOS	28
1.1.1	Objetivo Geral	28
1.1.2	Objetivos Específicos	29
1.2	ORGANIZAÇÃO DA DISSERTAÇÃO	29
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	CONCEITOS RELACIONADOS A ROBÓTICA MÓVEL	31
2.1.1	Representação de posição e orientação	32
2.1.2	Localização e mapeamento	35
2.2	TÉCNICAS DE LOCALIZAÇÃO PROBABILÍSTICAS ..	37
2.2.1	Localização de Markov	39
2.2.1.1	Atualização de previsão	40
2.2.1.2	Atualização de medição	40
2.2.2	Localização por filtro de Kalman	41
2.2.2.1	Atualização de previsão	42
2.2.2.2	Atualização de medição (correção)	42
2.2.3	Outros sistemas de localização	44
2.3	TÉCNICAS DE SLAM	45
2.3.1	Definição matemática de SLAM	45
2.3.2	EKF-SLAM	47
2.3.3	SLAM baseado em grafos	49
2.3.4	SLAM com filtro de partículas (FastSLAM)	50
2.4	CONCEITOS RELACIONADOS À VISÃO COMPUTACIONAL	52
2.4.1	Detectores e descritores de características locais ..	52
2.4.1.1	SIFT	52
2.4.1.2	SURF	55
2.4.1.3	BRIEF	57
2.4.1.4	ORB	58
2.4.1.5	BRISK	59
2.4.1.6	FREAK	62
2.4.1.7	GFTT	64
2.4.2	RANSAC	64
2.4.3	Registro de imagens	65
2.5	CONCLUSÃO	69
3	TRABALHOS RELACIONADOS	71
3.1	LOCALIZAÇÃO	71

3.2	SLAM E REGISTRO 3D	73
3.3	AVALIAÇÃO DE DETECTORES E DESCRIPTORES DE CARACTERÍSTICAS LOCAIS	74
3.4	CONCLUSÃO	76
4	DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA REMOTO DE LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS	77
4.1	SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO	77
4.1.1	Análise de requisitos	77
4.1.1.1	Requisitos funcionais	77
4.1.1.2	Requisitos não-funcionais	78
4.1.2	Recursos	79
4.1.2.1	Software	79
4.1.2.2	Hardware	81
4.1.3	Avaliação Experimental	84
4.1.4	Estrutura do algoritmo	86
4.1.5	Critérios de Avaliação	88
5	EXPERIMENTOS E RESULTADOS	91
5.1	CONFIGURAÇÃO ALTERNATIVA	96
6	CONSIDERAÇÕES FINAIS	101
6.1	CONCLUSÃO	101
6.2	TRABALHOS FUTUROS	102
	REFERÊNCIAS	103
	APÊNDICE A – Árvore de quadros	113

1 INTRODUÇÃO

A robótica é uma área da ciência multidisciplinar, na qual ocorre um expressivo desenvolvimento em diversas áreas de pesquisa. Como observado por Holtkamp e Jong (2006), a maior diversidade e crescimento de suas aplicações sugerem, cada vez mais, a importância deste tema em aplicações de sistemas produtivos, de serviços e do lazer.

Robôs são dispositivos mecânicos versáteis equipados com sensores e atuadores sob o controle de um sistema computacional. Desta forma, eles são capazes de realizar tarefas executando movimentos em um espaço físico com objetos (SACCHETIN, 2006). De acordo com Gaspar (1994), um robô móvel consiste em uma plataforma móvel sobre a qual é integrada percepção e ação no ambiente que o rodeia. A ação executada por atuadores guia o robô durante seu movimento pelo mundo. A percepção recolhe os dados dos sensores, interpreta-os de modo a melhorar a compreensão do mundo e propor ações através de controle.

Aplicações modernas exigem robôs móveis autônomos (AMR) que incluem uma variedade de sensores e uma alta capacidade computacional de modo que possibilitem essa autonomia do robô. Um requisito básico de qualquer entidade autônoma é a capacidade de agir por conta própria. No caso dos robôs móveis autônomos, é um requisito fundamental que eles consigam se localizar e obter uma representação do ambiente no qual se encontram, de modo que seja possível a navegação e realização das suas atividades.

Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), para um robô móvel autônomo, a navegação é a tarefa de deslocar-se entre pontos diferentes do ambiente com segurança. O problema geral de navegação pode ser dividido em quatro blocos, sendo que o sucesso na operação depende do sucesso em cada um deles:

- Percepção. O robô deve interpretar seus sensores para extrair dados relevantes e conseguir se localizar em relação a posições passadas.
- Localização. O robô deve determinar sua posição no ambiente. Na maioria das aplicações, o robô deve possuir ou construir uma representação de seu ambiente, esta tarefa é denominada mapeamento.
- Cognição. O robô deve decidir como agir para atingir seus objetivos. A fim de cumprir alguma tarefa, o robô deve saber para

onde está indo. Trata-se de reconhecer uma meta e decidir como chegar lá. A tarefa de encontrar um caminho para chegar à meta é conhecido como planejamento de trajetória.

- Controle de movimento. O robô deve modular suas saídas de motor para seguir a trajetória desejada e alcançar sua meta.

O problema de localização robótica pode ser classificado em três tipos de problemas: Rastreamento de posição (ou localização local), localização global e o problema do robô sequestrado. A localização local do robô é atualizada com base na conhecimento de sua posição anterior (rastreamento). Isso implica que a localização inicial do robô deve ser conhecida. A localização global, em contrapartida, pressupõe que a localização inicial do robô é desconhecida. Isso significa que o robô pode ser colocado em qualquer lugar do ambiente, sem conhecimento disto, e é capaz de localizar-se globalmente dentro dele. O problema do robô sequestrado, no qual ele é movido involuntariamente, se assemelha ao problema de localização global somente se o robô perceber que foi sequestrado. A dificuldade surge quando o robô não sabe que ele foi movido para outro local e acredita que sabe onde está, mas na verdade não sabe (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

A localização é uma das características mais importantes para a navegação autônoma de robôs móveis, assim, robustez e precisão são fatores fundamentais. Em geral, essa tarefa baseia-se no mapa do ambiente em questão. Em contrapartida, a construção do mapa do local depende da localização precisa do robô. Em ambientes desconhecidos este é um processo conflitante, mas correlacionado, conhecido como localização e mapeamento simultâneos (SLAM) (LIN et al., 2012).

Uma grande variedade de sensores vêm sendo utilizados para realizar a tarefa de localização (BICCHI et al., 2004), como o sistema de posicionamento global (GPS), *laser rangefinder* (LRF), unidade de medida inercial (IMU), entre outros que, ocasionalmente, podem ser caros e sempre têm restrições, erros e ruídos de medição. Assim como o ser humano, que usa sua visão para localizar-se e reconhecer o ambiente, robôs móveis podem tirar proveito de recursos visuais que podem ser facilmente integrados em plataformas multisensoriais. Através da utilização de câmeras e visão computacional, robôs móveis são capazes de obter informações mais detalhadas do ambiente, aumentando a versatilidade e segurança nas operações do robô, como detecção e localização de objetos ou obstáculos, úteis no desempenho de tarefas de manipulação ou navegação, por exemplo. Por outro lado, se faz necessário um maior poder computacional para lidar com tantos detalhes, o que pode

não ser viável em aplicações mais simples. Valores de odometria visual são, em geral, obtidos através da relação entre características semelhantes, detectadas entre 2 imagens consecutivas, essa relação é computada através de técnicas denominadas *Structure from Motion* (SfM).

Como cada sensor, por mais preciso que seja, possui suas limitações e erros de medição, sistemas que fazem uso de múltiplos sensores podem empregar técnicas de fusão temporal ou multissensorial e/ou técnicas probabilísticas para obtenção de valores cada vez mais precisos.

Outro problema de grande importância no desenvolvimento de robôs móveis verdadeiramente autônomos é o mapeamento, que tem aplicações em navegação, manipulação, mapeamento semântico e telepresença (HENRY et al., 2012). Mapeamento robótico consiste na aquisição de modelos espaciais dos ambientes físicos através de robôs móveis e, para tanto, os robôs devem possuir sensores que lhe permitam perceber o mundo exterior. No entanto, esses sensores também estão sujeitos a erros, muitas vezes referidos como ruído de medição. Além disto, a maioria dos sensores estão sujeitos a limitações de alcance que fazem com que seja necessário que um robô navegue através de seu ambiente para construção de uma representação da sua área de trabalho (THRUN et al., 2002).

Embora funcionalidades autônomas inteligentes dos robôs desempenhem um papel importante, especialmente para o controle de locomoção, ainda existe uma quantidade significativa de controle humano em forma de teleoperação necessária para a operação dos sistemas. Ao fazer isso, é de extrema importância que os operadores tenham uma percepção da situação através de representações em 3D do ambiente (VASKEVICIUS et al., 2010).

Sensores de alcance de baixo custo são uma alternativa atraente aos caros sensores LRF em tarefas como mapeamento ou representação 2D de interiores, vigilância, robótica e forense (KHOSHELHAM; EL-BERINK, 2012). Câmeras RGB-D, como o Microsoft Kinect desenvolvido pela Prime-Sense e Microsoft, proporcionam informações de cores e profundidade por pixel em imagens sincronizadas e em tempo real a um preço acessível. O processo de captação consiste em obter uma imagem colorida (RGB) e realizar uma medição de profundidade (D) com técnica de luz estruturada (CRUZ; LUCIO; VELHO, 2012). Usando um sensor de profundidade, essas câmeras evitam a complexidade da computação visual correspondente para estimativa de profundidade em combinações de imagens de câmeras estéreo. Assim, sensores deste tipo vêm ganhando destaque em pesquisas relacionadas a localização e

mapeamento.

A navegação robótica baseada em visão tem sido um tema bem explorado por pesquisadores tanto na área de robótica como na área de visão computacional (NIN; OSÓRIO, 2011). As diferentes estratégias de navegação visual propostas na literatura fazem uso de várias configurações para obter a informação sobre o ambiente necessária para navegar. A maioria das configurações é baseada em sistemas monoculares, binoculares (câmeras estéreo) e câmeras omnidirecionais, que possuem uma visão 360° do ambiente geralmente obtida através de um espelho convexo cônico, esférico, parabólico ou hiperbólico. Com câmeras omnidirecionais é mais fácil encontrar e rastrear pontos de referência, uma vez que amplia-se o campo de visão do robô (BONIN-FONT; ORTIZ; OLIVER, 2008).

A introdução de visão em um robô móvel implica em um significativo aumento das suas capacidades sensoriais e, portanto, em um correspondente aumento na versatilidade e segurança na operação do robô. Da visão robótica, esperam-se soluções ou simplificações para problemas de detecção e localização de objetos ou obstáculos, úteis no desempenho de tarefas de manipulação ou navegação, por exemplo. Por outro lado, como observado por Hadda e Knani (2013), as imagens geralmente contêm uma abundante quantidade de informação, algumas das quais podem não ser úteis para a tarefa que o robô desempenha, requerendo assim algum tipo de filtragem e, portanto, o tempo necessário para compreender e converter essas informações em dados úteis pode ser elevado.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é determinar os problemas envolvidos na localização, mapeamento e reconstrução 3D de ambientes para robôs móveis através de métodos baseados em visão computacional e fusão de sensores e, posteriormente, propor um sistema de localização probabilístico que realiza a fusão de dados de odometria visual, odometria dos *encoders* e a pose do robô obtida através do escaneamento de um laser 2D para a obtenção de uma pose absoluta mais precisa e robusta que será utilizada para o mapeamento 3D representado por nuvem de pontos (*point cloud*). Em seguida, analisar este sistema em um cenário real, usando como plataforma um robô móvel (Pioneer 3-DX)

montado com os sensores em questão para coleta dos dados, que foram processados remotamente em um computador para localização do robô e mapeamento do ambiente simultaneamente em tempo real utilizando o ROS (*Robot Operating System*). Assim, este trabalho possui enfoque em técnicas de visão computacional e fusão de sensores capazes de acomodar as necessidades da robótica móvel.

1.1.2 Objetivos Específicos

Os seguintes objetivos específicos foram projetados para atingir o objetivo geral:

- Investigar artigos e implementações semelhantes na literatura.
- Desenvolver um sistema de localização probabilístico do robô, baseado em um filtro de Kalman estendido, registrando dados obtidos pelos sensores e simulando o comportamento do sistema com a fusão destes dados e a estimativa da pose absoluta do robô.
- Implementar o sistema de localização com um mapeamento e reconstrução 3D baseada em nuvem de pontos na forma de registro (*point cloud*).
- Implementar este sistema em um cenário real, usando como plataforma um robô móvel (Pioneer 3-DX) montado com os sensores em questão para coleta dos dados, e realizar o processo de SLAM remotamente (teleoperação) em um ambiente interno, plano e com boa luminosidade.
- Avaliar o desempenho do sistema proposto com base em critérios de avaliação a serem estudados.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

A fundamentação teórica utilizada para o desenvolvimento deste trabalho é exposta no capítulo dois. O capítulo três apresenta uma revisão bibliográfica, destacando o estado da arte na área de pesquisa em que este trabalho se encontra.

A construção do sistema de localização e mapeamento remotos, incluindo as respectivas características, requisitos, recursos e estruturas de funcionamento é apresentada no capítulo quatro.

O capítulo cinco exibe os experimentos realizados com o sistema desenvolvido, bem como os resultados e suas análises de acordo com os respectivos critérios de avaliação.

O capítulo seis expõe as considerações finais desta dissertação, as conclusões e sugestões para desenvolvimentos futuros, visando a ampliação e o contínuo aprimoramento do trabalho desenvolvido.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos que constituem a base teórica deste trabalho. Serão expostas definições relacionadas a robótica móvel, técnicas probabilísticas de localização e mapeamento e visão computacional, abordando temas como a representação de posição e orientação de robôs móveis, propagação de erros em medidas, fusão de sensores, detectores e descritores de características locais, tipos de mapeamento e registro, bem como a teoria necessária para realizar estas operações.

2.1 CONCEITOS RELACIONADOS A ROBÓTICA MÓVEL

A robótica alcançou grande sucesso no mundo da produção industrial. Braços robóticos, ou manipuladores, compreendem uma indústria de dois bilhões de dólares. No entanto, apesar de todos o sucesso, esses robôs comerciais sofrem de uma desvantagem, que é a falta de mobilidade. Um manipulador fixo tem uma amplitude limitada de movimento que depende de onde ele é instalado (parafusado). Em contraste, um robô móvel é capaz de viajar por toda a fábrica, aplicando de forma flexível as suas habilidades onde quer que seja mais eficaz (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011). Além disto, eles podem operar em ambientes hostis ou inóspitos, como é o caso de robôs enviados a Chernobyl após o acidente nuclear ou do robô *Curiosity*, mostrado na Figura 1, enviado à marte.

Outros robôs móveis comerciais operam não somente em ambientes inóspitos, mas dividem espaço com humanos em seus ambientes. Esses robôs são atraentes não apenas pela mobilidade, mas por causa de sua autonomia, e assim a sua capacidade de manter um senso de posição e navegar sem intervenção humana é primordial (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

Um robô móvel precisa de mecanismos de locomoção que lhe permitam mover-se livremente pelo ambiente e há uma grande variedade de mecanismos capazes de gerar movimento, geralmente os mais usados são mecanismos de rodas (tecnologia amplamente utilizada em veículos humanos), usando pernas articuladas (a mais simples das abordagens biológicas para a locomoção) ou hélices (e.g. em UAVs ou veículos subaquáticos).

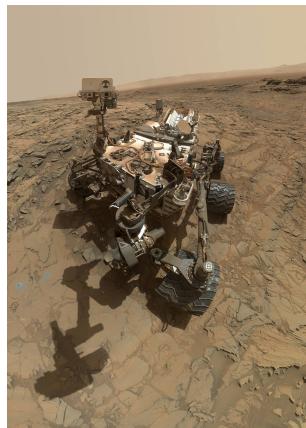


Figura 1 – Autorretrato do robô Curiosity da NASA em Marte.
Fonte: NASA/JPL-Caltech/MSSS.¹

2.1.1 Representação de posição e orientação

A posição e orientação de um corpo rígido no espaço compõe a denominada pose deste corpo. A cinemática do robô descreve a pose, velocidade, aceleração e todos os derivados de ordem superior da pose dos organismos que integram um mecanismo (SICILIANO; KHATIB, 2008). A cinemática é o estudo mais básico de como os sistemas mecânicos se comportam. Na robótica móvel, é preciso entender o comportamento mecânico do robô tanto para projetar apropriadamente robôs móveis para tarefas quanto para entender como criar software de controle para uma instância de hardware de robô móvel (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

Para obter uma representação da pose do robô, ele é modelado como um corpo rígido sobre rodas, operando em um plano horizontal. A dimensionalidade total deste chassi robótico no plano é três, dois para posição no plano e um para orientação ao longo do eixo vertical, que é ortogonal ao plano. Claro, existem graus adicionais de liberdade e flexibilidade devido aos tipos de rodas e articulações de rodas. No entanto, refere-se apenas ao corpo rígido do robô, ignorando as articulações e graus de liberdade interna ao robô e suas rodas (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

¹Disponível em: <<https://www.nasa.gov/image-feature/jpl/pia19920/curiosity-self-portrait-at-big-sky-drilling-site>>. Acessado em Abr. 2018.

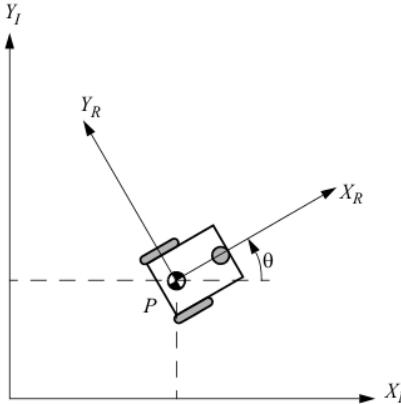


Figura 2 – O quadro de referência global (X_I, Y_I) e o quadro de referência (X_R, Y_R) local do robô em 2D (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

Um sistema de coordenadas i consiste de uma origem, denotada O_i , e uma triade de vetores ortogonais entre si na base, denotados $(x_i, y_i$ e $z_i)$, que são todos fixos dentro de um corpo particular. A pose de um corpo será sempre expressa em relação a alguma outra pose, de modo que esta pose possa ser expressa como a pose de um coordenada relativa a outra. Da mesma forma, deslocamentos de corpos rígidos podem ser expressos como deslocamentos entre dois sistemas de coordenadas, um dos quais pode ser referido como em movimento, enquanto o outro pode ser referido como fixo. Isto indica que o observador está localizado em uma posição fixa no interior do sistema de referência fixo (SICILIANO; KHATIB, 2008).

Para especificar a posição do robô no plano, é necessário estabelecer uma relação entre os sistemas de referência global e local do robô. A posição de origem do sistema de coordenadas i relativa ao sistema de coordenadas j pode ser indicada pelo seguinte vetor 3 X 1:

$${}^j p_i = \begin{pmatrix} {}^j p_i^x \\ {}^j p_i^y \\ {}^j p_i^z \end{pmatrix} \quad (2.1)$$

As componentes deste vetor são as coordenadas cartesianas de O_i no sistema de coordenadas j , que são as projeções do vetor ${}^j p_i$ sobre

os eixos correspondentes. Os componentes do vetor também poderiam ser expressos como coordenadas esféricas ou cilíndricas de O_i em j . Tais representações têm vantagens para a análise de mecanismos robóticos incluindo juntas esféricas e cilíndricas (SICILIANO; KHATIB, 2008).

Uma translação é um deslocamento no qual nenhum ponto no corpo rígido permanece em sua posição inicial e todas as linhas retas no corpo rígido permanecem paralelas à sua orientação inicial. A translação de um corpo no espaço pode ser representada pela combinação das posições de antes e depois da translação. Por outro lado, a posição de um corpo no espaço pode ser representada como uma translação que leva o corpo de uma posição na qual o sistema de coordenadas fixo ao corpo do robô coincide com o sistema de coordenadas fixo para a posição atual na qual os dois sistemas não coincidem. Portanto, qualquer representação de posição pode ser usada para criar uma representação de deslocamento e vice-versa (SICILIANO; KHATIB, 2008).

Uma rotação é um deslocamento no qual pelo menos um ponto do corpo rígido permanece na sua posição inicial e nem todas as linhas no corpo permanecem paralelas às suas orientações iniciais. Por exemplo, um corpo em uma órbita circular gira em torno de um eixo através do centro de sua trajetória circular, e cada ponto no eixo de rotação é um ponto no corpo que permanece em sua posição inicial. Como no caso de posição e translação, qualquer representação de orientação pode ser usada para criar uma representação de rotação e vice-versa (SICILIANO; KHATIB, 2008).

Existe uma variedade significativamente maior de representações de orientação do que de representações de posição. Alguns exemplos são ângulos de Euler, ângulo/eixo, quatérnions e matrizes de rotação.

A representação por ângulos de Euler é composta por três ângulos: Rotação em torno do eixo X, denominada ângulo de rolagem (α); Rotação em torno do eixo Y, denominada ângulo de arfagem (β); e rotação em torno do eixo Z, denominada ângulo de guinada (γ).

Um quatérnion pode ser visto como um vetor de 4 componentes, composto de um vetor escalar e um ordinário, ou como um número complexo com três partes imaginárias diferentes (HORN, 1987).

Segundo (HORN, 1987), um quatérnion q é definido com a forma:

$$q = q_0 + q_1 i + q_2 j + q_3 k \quad (2.2)$$

onde os componentes q_0, q_1, q_2, q_3 são escalares, com q_0 representando a parte real e os outros termos as três partes imaginárias, usualmente referidos como parâmetros de Euler, e i, j e k são operado-

res. Os operadores são definidos de modo que se satisfaçam as seguintes regras combinatórias:

$$\begin{aligned} ii &= jj = kk = -1, \\ ij &= k, jk = i, ki = j, \\ ji &= -k, kj = -i, ik = -j. \end{aligned} \quad (2.3)$$

Em várias ocasiões, q_0 é referido como a parte escalar do quatérnion, e $(q_1, q_2, q_3)^T$ como a parte vetorial. Um quatérnion unitário pode então ser definido tal que $q\tilde{q} = 1$ (SICILIANO; KHATIB, 2008). De acordo com Siciliano e Khatib (2008), quatérnions unitários são usados para descrever a orientação, sendo extremamente úteis para problemas em robótica que resultam em singularidades representacionais na notação vetor/matriz. Um vetor é definido na notação quatérnion como um quatérnion com $q_0 = 0$. Assim, um vetor $p = (p_x \ p_y \ p_z)^T$ pode ser expresso como um quatérnion $p = p_x i + p_y j + p_z k$.

Para qualquer quatérnion unitário q , a operação $qp\tilde{q}$ executa uma rotação do vetor p em torno da direção $(q_1 \ q_2 \ q_3)^T$. Isto pode ser observado expandindo a operação $qp\tilde{q}$ e comparando os resultados com a matriz de rotação equivalente.

Além disto, Siciliano e Khatib (2008) afirmam que os quatérnions unitários estão intimamente relacionados com a representação eixo angular de orientação, onde um ângulo único θ em combinação com um vetor unitário w pode também indicar a orientação do sistema de coordenadas i em relação ao sistema de coordenadas j . Neste caso, q_0 corresponde ao angulo de rotação, enquanto q_1, q_2 e q_3 definem o eixo de rotação.

2.1.2 Localização e mapeamento

A localização é o processo de inferir a pose do robô dentro de um mapa (FILLIAT; MEYER, 2003). O mapeamento robótico aborda o problema de aquisição de modelos espaciais de ambientes físicos através de robôs móveis e é comumente tratado como um dos problemas mais importantes na busca da construção de robôs móveis verdadeiramente autônomos (THRUN et al., 2002).

O problema de representar o ambiente em que o robô se move é dual ao problema de representar a posição, ou posições possíveis, do robô no ambiente. As decisões tomadas em relação à representação ambiental podem ter impacto nas opções disponíveis para a representação da posição do robô. Muitas vezes a fidelidade da representação de posi-

ção é limitada pela fidelidade do mapa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

As abordagens existentes podem ser agrupadas em três paradigmas diferentes: topológico, métrico e híbrido. Abordagens no paradigma métrico capturam as propriedades geométricas do ambiente do robô. Por outro lado, as abordagens no paradigma topológico descrevem a conectividade de diferentes locais, em geral representando ambientes como uma lista de lugares significativos conectados via arcos. O modelo híbrido permite o uso das outras abordagens de modo complementar (THRUN et al., 1998) (TOMATIS, 2001).

Siegwart, Nourbakhsh e Scaramuzza (2011) afirmam que se alguém pudesse anexar um sensor GPS (sistema de posicionamento global) preciso à um robô móvel, evitaria grande parte do problema de localização. O GPS informaria a posição exata do robô, em ambientes internos ou externos, de modo que a pergunta “Onde estou?” teria sempre uma resposta imediatamente disponível. Infelizmente, tal sensor não é atualmente prático. A rede GPS existente fornece precisão dentro de vários metros, o que é inaceitável para a localização de robôs móveis em escala humana, assim como robôs móveis em miniatura, por exemplo robôs de mesa e nanorrobôs futuristas que podem navegar dentro de corpos. Além disso, as tecnologias de GPS podem não funcionar em ambientes fechados ou em áreas obstruídas e, portanto, são limitadas em seu espaço de trabalho. Também não funcionariam em outros planetas devido à falta dos satélites GPS, como é o caso dos robôs enviados à Marte.

Os sensores e atuadores do robô desempenham um papel integral em todas as formas de localização. É por consequência da imprecisão e incompletude desses sensores e atuadores que a localização apresenta desafios tão difíceis. Os sensores são a entrada fundamental do robô para o processo de percepção e, portanto, o grau em que os sensores podem discriminar o estado do mundo é crítico. O ruído do sensor induz uma limitação na consistência das leituras do sensor, reduzindo o conteúdo de informação útil das leituras do sensor. Uma solução é levar em consideração múltiplas leituras, empregando fusão temporal ou fusão multissensor para aumentar o conteúdo de informação geral das entradas do robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.2 TÉCNICAS DE LOCALIZAÇÃO PROBABILÍSTICAS

Como visto anteriormente, em situações reais os sensores e atuadores de robôs móveis estão sujeitos a imperfeições que resultam em imprecisões e incompletudes nas suas operações. O ruído dos sensores induz uma limitação na consistência das leituras dos sensores no mesmo estado ambiental e, consequentemente, no número de bits úteis disponíveis a partir de cada leitura do sensor. Outra deficiência dos sensores robóticos móveis, a não exclusividade das leituras dos sensores, faz com que eles produzam pouco conteúdo de informação e pode, por exemplo, fazer com que o robô se confunda entre lugares distintos porém semelhantes, exacerbando ainda mais o problema da percepção e, portanto, da localização. Além disto, os atuadores também sofrem ruídos, uma única ação tomada por um robô móvel pode levar a vários resultados possíveis diferentes, embora, do ponto de vista do robô, o estado inicial antes da ação ser tomada seja bem conhecido. Portanto, os atuadores de robôs móveis introduzem incerteza sobre o estado futuro, o simples ato de mover tende a aumentar a incerteza de um robô móvel (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), na odometria (apenas sensores das rodas) e no *dead reckoning* (também nos sensores de rumo), a atualização da posição é baseada em sensores proprioceptivos. O movimento do robô, detectado com *encoders* de roda ou sensores de rumo ou ambos, é integrado para calcular a posição. Como os erros de medição do sensor são integrados, o erro de posição se acumula ao longo do tempo. Assim, a posição tem que ser atualizada de tempos em tempos por outros mecanismos de localização. Caso contrário, o robô não conseguirá manter uma estimativa de posição significativa no longo prazo.

Um modelo de erro para estimativa de posição odométrica pode ser obtido, de modo a ser útil na aplicação de técnicas de localização onde não é considerada somente a leitura dos sensores mas também os erros estimados em cada leitura contidos em uma matriz de covariâncias. A Figura 3 apresenta um exemplo de evolução na incerteza da posição de um robô móvel de acionamento diferencial (duas rodas paralelas acionadas independentemente e uma roda castor sem acionamento) em duas situações, movimento linear e circular. Nota-se que a incerteza perpendicular ao movimento cresce muito mais rápido do que na direção do movimento. Isso é resultado da integração da incerteza sobre a orientação do robô. As elipses desenhadas em torno das posições do robô representam as incertezas na direção x, y. Embora a incerteza

da orientação θ não ser representada na figura, seu efeito de ser indiretamente observado pois influencia diretamente os valores de y , já que o robô tem acionamento diferencial (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

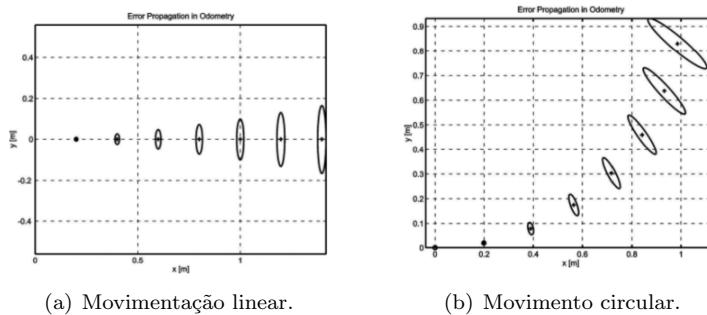


Figura 3 – Propagação de erro na posição de um robô móvel diferencial (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

A navegação de um robô móvel pode ser baseada em comportamentos ou em mapas. No primeiro caso, o robô pode ser programado, por exemplo, para seguir uma parede como forma de se deslocar entre dois pontos sem ter que calcular um plano de trajetória propriamente dito. Em contrapartida, a abordagem baseada em mapas inclui ambas etapas de localização e cognição, o robô tenta se localizar explicitamente pela coleta de dados dos sensores e, então, a atualização de uma convicção sobre a posição em relação a um mapa do ambiente. As grandes vantagens das abordagens baseadas em mapas para navegação são (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011):

- O conceito de posição baseado em mapas torna a convicção do sistema sobre a posição transparentemente disponível para operadores humanos.
- A existência de um mapa, por sua vez, representa uma mediação para comunicação entre operador e robô, o operador pode disponibilizar um novo mapa ao robô caso ele mude de ambiente.
- O mapa, se criado pelo robô, pode ser usado também por humanos para análise do ambiente.

A questão fundamental que diferencia vários sistemas de localização baseados em mapas é a questão da representação. O robô precisa de uma representação do mapa, assim como uma representação

de sua crença em relação a sua posição no mapa. A combinação de diferentes formas de representação resulta nos mais variados níveis de complexidade estrutural, computacional e na precisão total da localização. Com relação a representação de sua crença da posição no mapa, o robô pode ter uma única hipótese singular ou múltiplas hipóteses que, neste caso, da ao robô a habilidade de manter um senso de posição enquanto anota explicitamente a incerteza do robô sobre sua própria posição (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Idealmente, o estado de crença do robô mudará, ao longo do tempo, conforme for consistente com suas saídas de motor e entradas perceptuais. Porém alguns métodos não fornecem nenhuma indicação das chances relativas entre várias posições possíveis do robô. Técnicas de localização probabilística baseadas em mapas diferem disso porque identificam explicitamente probabilidades com as possíveis posições do robô e, por essa razão, esses métodos têm sido o foco de pesquisas recentes. A razão das abordagens probabilísticas para localização de robôs móveis terem sido desenvolvidas é que os dados provenientes dos sensores do robô são afetados por erros de medição e, portanto, só podemos calcular a probabilidade de o robô estar em uma determinada posição. Esta nova área de pesquisa é chamada de robótica probabilística. A ideia chave em robótica probabilística é representar a incerteza usando teoria de probabilidade (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Nas seções a seguir, são apresentadas duas técnicas Bayesianas de localização probabilística baseada em mapas, localização de Markov e localização por filtro de Kalman.

2.2.1 Localização de Markov

A localização de Markov acompanha o estado de crença do robô usando uma função de densidade de probabilidade arbitrária para representar a posição do robô. Na prática, todos os sistemas de localização de Markov conhecidos implementam essa representação de crença genérica colocando primeiro o espaço de configuração do robô (x, y, θ) em um número finito e discreto de poses possíveis do robô no mapa. Em aplicações reais, o número de poses possíveis pode chegar a grandes escalas (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Como afirmado por Siegwart, Nourbakhsh e Scaramuzza (2011), o processo de localização probabilística do robô consiste na iteração das atualizações de previsão e medição. Eles calculam o estado de crença que resulta quando novas informações (por exemplo, valores de *encoders* e

dados de medição) são incorporadas em um estado de crença anterior com densidade de probabilidade arbitrária. Tanto na localização de filtro de Markov quanto de Kalman, a atualização de previsão é baseada no teorema da probabilidade total, enquanto a atualização de medição é baseada na regra de Bayes.

2.2.1.1 Atualização de previsão

Pelo teorema total de probabilidade, é possível o cálculo da crença atual do robô ($\overline{bel}(x_t)$) como uma função da crença anterior ($bel(x_{t-1})$) e os dados proprioceptivos (u_t), por exemplo medição dos encoders ou entrada de controle (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011):

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx_{t-1} \quad (\text{caso contínuo}) \quad (2.4)$$

$$\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})bel(x_{t-1}) \quad (\text{caso discreto}) \quad (2.5)$$

Observa-se que, nesta etapa, a crença atual do robô é computada sobre todas as posições possíveis do robô x_t , com a integral no caso contínuo e a somatória no caso discreto. Isso significa que, em situações reais, onde o número de células usadas para representar o robô pode ser de vários milhões, os cálculos podem se tornar impraticáveis e, portanto, impedir a operação em tempo real.

2.2.1.2 Atualização de medição

Nesta etapa, o robô corrige sua posição anterior combinando-a oportunamente com a informação de seus sensores exteroceptivos. É utilizada a regra de Bayes para calcular o novo estado de crença do robô ($bel(x_t)$) como uma função de seus dados de medição (z_t) e seu antigo estado de crença ($\overline{bel}(x_t)$) (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011):

$$bel(x_t) = \eta p(z_t|x_t, M)\overline{bel}(x_t), \quad (2.6)$$

Onde, $p(z_t|x_t, M)$ é o modelo de medição probabilística, isto é, a

probabilidade de observar os dados de medição z_t dado o conhecimento do mapa M e a posição do robô x_t . Portanto, o novo estado de crença é simplesmente o produto entre o modelo de medição probabilístico e o estado de crença anterior. Observe que a equação 2.5 não atualiza apenas uma pose, mas todas as poses possíveis x_t .

2.2.2 Localização por filtro de Kalman

O modelo de localização de Markov pode representar qualquer função de densidade de probabilidade arbitrária sobre a posição do robô. Esta abordagem é muito geral, mas, devido à sua generalidade, ineficiente. Considere, em vez disso, as principais demandas de um sistema de localização de robôs. Pode-se argumentar que não é a replicação exata de uma curva de densidade de probabilidade, mas o problema de fusão de sensores que é a chave para uma localização robusta. Os robôs geralmente incluem um grande número de sensores heterogêneos, cada um fornecendo pistas sobre a posição do robô e, criticamente, cada um sofrendo de suas próprias falhas e imprecisões. A localização ideal deve levar em conta as informações fornecidas por todos esses sensores. Uma técnica poderosa para alcançar essa fusão de sensores é denominada filtro de Kalman. Esse mecanismo é, de fato, mais eficiente que a localização de Markov devido à simplificações importantes ao representar a função de densidade de probabilidade do estado de crença do robô e até mesmo suas leituras de sensores individuais. O benefício dessas simplificações é um algoritmo de processamento de dados recursivo e ótimo resultante. Ele incorpora todas as informações, independentemente da precisão, para estimar o valor atual da variável de interesse (ou seja, a posição do robô) (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

O filtro de Kalman é um mecanismo matemático que produz uma estimativa ótima do estado do sistema com base no conhecimento do sistema e do dispositivo de medição, na descrição do ruído do sistema e nos erros de medição e na incerteza nos modelos de dinâmica. Assim, o filtro de Kalman funde os sinais dos sensores e o conhecimento do sistema de uma maneira otimizada. Dentro da teoria do filtro de Kalman, o sistema é considerado linear e com ruído gaussiano branco, mas na maioria das aplicações de robôs móveis, o sistema não é linear. Nesses casos, o filtro de Kalman é geralmente aplicado após a linearização do sistema. A extensão do filtro de Kalman para sistemas não lineares é conhecida como Filtro de Kalman Estendido (EKF), mas sua otimiza-

ção não pode ser garantida. A suposição de erro gaussiano pode não ser válida para aplicações de robôs móveis, mas mesmo assim os resultados são extremamente úteis. Em outras disciplinas de engenharia, a suposição de erro gaussiana, em alguns casos, mostrou-se bastante precisa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Durante as atualizações de previsão e medição, apenas a média (μ_t) e a matriz de covariâncias (Σ_t) são atualizadas. Portanto, o filtro de Kalman é baseado em quatro equações: duas para atualização μ_t e Σ_t na atualização de previsão e outras duas na atualização de medição. Na filtragem de Kalman, a atualização de medição também é comumente chamada de atualização de correção.

2.2.2.1 Atualização de previsão

A posição do robô \hat{x}_t no intervalo de tempo t é prevista com base em sua localização anterior no tempo $t - 1$ e seu movimento devido à entrada de controle u_t :

$$\hat{x}_t = f(x_{t-1}, u_t). \quad (2.7)$$

Conhecendo a planta e o modelo de erro, também podemos calcular a variância \hat{P}_t associada a esta predição usando a equação derivada do teorema da probabilidade total aplicada às distribuições gaussianas (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011):

$$\hat{P}_t = F_x \cdot P_{t-1} \cdot F_x^T \cdot Q_t \cdot F_u^T, \quad (2.8)$$

Onde P_{t-1} é a matriz de covariâncias de erro do robô no estado anterior x_{t-1} e Q_t é a covariância do ruído associada ao modelo de movimento.

2.2.2.2 Atualização de medição (correção)

De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), esta etapa é realizada em 4 passos, são eles:

1. Observação: O primeiro passo é obter as medições reais dos sensores z_t no tempo t .
2. Previsão de medição: Com base em sua posição prevista no mapa, o robô gera uma previsão de medição que consiste nas caracterís-

ticas que o robô espera observar a partir da posição em que pensa estar (por exemplo, a posição estimada no passo de previsão).

3. Combinação: Nesta etapa é computada a melhor combinação entre características extraídas durante a observação e as características esperadas que foram selecionadas na previsão de medição.
4. Estimação: Por fim, o filtro de Kalman funde as informações provindas de todas as combinações para atualizar o estado de crença do robô.

A etapa de estimação aplica a regra de Bayes, calculando a melhor estimativa x_t da posição do robô baseada na previsão de posição \hat{x}_t e todas as observações z_t^i no tempo t . Pela aplicação da regra de Bayes em distribuições gaussianas, pode-se atualizar a estimativa de posição x_t e sua matriz de covariância associada P_t por:

$$x_t = \hat{x}_t + K_t v_t, \quad (2.9)$$

$$P_t = \hat{P}_t - K_t \cdot \Sigma_{IN_t} \cdot K_t^T, \quad (2.10)$$

sendo o ganho de Kalman:

$$K_t = \hat{P}_t \cdot H_t^T \cdot (\Sigma_{IN_t})^{-1} \quad (2.11)$$

Se for imposta uma matriz identidade em H , as equações (2.9) e (2.10) são simplificadas:

$$x_t = \hat{x}_t + \hat{P}_t (\hat{P}_t + R_t)^{-1} (z_t - \hat{x}_t) \quad (2.12)$$

$$P_t = \hat{P}_t - \hat{P}_t (\hat{P}_t + R_t)^{-1} \hat{P}_t \quad (2.13)$$

A nova estimativa fundida da posição do robô é novamente submetida a uma curva de densidade de probabilidade gaussiana. Sua média e covariância são simplesmente funções de duas entradas, média e covariância. Assim, o filtro de Kalman proporciona uma representação compacta e simplificada da incerteza e é uma técnica extremamente eficiente para combinar estimativas heterogêneas para gerar uma nova estimativa para a posição do nosso robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.2.3 Outros sistemas de localização

De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), a localização de Markov e a localização por filtro de Kalman têm sido duas estratégias extremamente populares em pesquisas de robôs móveis que navegam em ambientes internos. Elas são bem consolidadas, portanto, apresentam um comportamento bem definido. Mas existem outras técnicas de localização probabilísticas que foram usadas em plataformas de robôs móveis comerciais e de pesquisa. Algumas técnicas que merecem destaque são a localização por filtro de Kalman *Unscented* (UKF), a localização em grade e a localização de Monte Carlo. O UKF é semelhante ao EKF na medida em que também assume distribuições gaussianas, mas possui uma maneira diferente de linearizar os modelos de movimento e medições.

Segundo Aulinhas et al. (2008), o filtro de Kalman *Unscented* (UKF) aborda as questões de aproximação do EKF e os pressupostos de linearidade do filtro de Kalman. O filtro de Kalman é executado corretamente nos casos lineares e é aceito como um método eficiente para propagar analiticamente a variável aleatória Gaussiana (GRV) através de uma dinâmica de sistema linear. Para modelos não lineares, o EKF aproxima os termos ótimos linearizando as equações dinâmicas. O EKF pode ser visto como uma aproximação de primeira ordem para a solução ótima. Nestas aproximações a distribuição dos estados é aproximada por uma GRV, que é então propagada analiticamente através da linearização de primeira ordem do sistema não-linear. Essas aproximações podem introduzir grandes erros na verdadeira média e covariância posteriores, o que pode levar, às vezes, à divergência do filtro. No UKF, a distribuição do estado é novamente representada por um GRV, mas agora é especificada usando um conjunto mínimo de pontos de amostragem cuidadosamente escolhidos. Esses pontos de amostragem capturam completamente a média e covariância real do GRV e, quando propagados pelo sistema não-linear verdadeiro, capturam a média e covariância posteriores com precisão de até terceira ordem para qualquer não-linearidade. Para fazer isso, a transformação *Unscented* é usada.

Por outro lado, a localização em grade e a localização de Monte Carlo não se limitam à distribuições unimodais. Enquanto a localização em grade usa o chamado filtro de histograma para representar a crença do robô, a localização de Monte Carlo usa filtros de partículas. Este último é provavelmente o algoritmo de localização mais popular (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

2.3 TÉCNICAS DE SLAM

Enquanto a localização é o problema de estimar a posição do robô (consequentemente, sua trajetória) dado um mapa conhecido do ambiente, o mapeamento é a construção do mapa do ambiente, conhecendo o verdadeiro trajeto do robô. O objetivo do SLAM é recuperar o trajeto do robô e o mapa do ambiente usando apenas os dados coletados por seus sensores proprioceptivos e exteroceptivos. Esses dados são tipicamente o deslocamento do robô estimado a partir da odometria e características (por exemplo, cantos, linhas, planos) extraídas de um laser, sensor ultrassônico ou de imagens de câmeras (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

A dificuldade do SLAM se da pelo fato de ambas trajetória estimada e características extraídas serem corrompidas por ruído. Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), o problema geral de construção de mapas é análogo ao problema do ovo e da galinha. Para localização, o robô precisa saber onde as características estão, enquanto para mapeamento, o robô precisa saber onde ele está no mapa.

Nesta seção, é apresentada uma definição matemática do problema de SLAM, e são descritos os três principais métodos desenvolvidos nas últimas décadas para solucionar o problema de SLAM, que deram origem a diversos outros algoritmos, são eles: EKF-SLAM, SLAM baseado em grafos e SLAM com filtro de partículas.

2.3.1 Definição matemática de SLAM

As terminologias e a definição matemática de SLAM são apresentadas por Siegwart, Nourbakhsh e Scaramuzza (2011), a pose de um robô em dado momento t é definida por x_t . Assim, trajetória do robô é dada por:

$$X_T = \{x_0, x_1, x_2, \dots, x_T\}, \quad (2.14)$$

onde T pode ser infinito. Em SLAM, assume-se que a locação inicial do robô x_0 é conhecida, enquanto as outras posições são desconhecidas. O movimento do robô entre os tempos $t - 1$ e t . Tais dados podem provir de leituras de sensores proprioceptivos (por exemplo, dos *encoders* de roda do robô) ou das entradas de controle fornecidas aos motores. A sequência de movimentos relativos do robô pode então ser escrita como:

$$U_T = \{u_0, u_1, u_2, \dots, u_T\}. \quad (2.15)$$

M representa o mapa verdadeiro do ambiente:

$$M = \{m_0, m_1, m_2, \dots, m_T\}, \quad (2.16)$$

sendo $m_i, i = 0 \dots n - 1$, vetores representando a posição do pontos de referência no mapa, que podem ser pontos, linhas, planos ou qualquer tipo de característica de alto nível (por exemplo, portas). Para efeito de simplificação, o mapa é considerado estático.

Enfim, considerando que o robô realiza uma medida a cada tempo, podemos denotar a sequência de observações de pontos de referência no quadro de referência do sensor conectado ao robô:

$$Z_T = \{z_0, z_1, z_2, \dots, z_T\}, \quad (2.17)$$

Por exemplo, se o robô é equipado com uma câmera a bordo, a observação z_i pode ser um vetor representando as coordenadas de um canto ou as de uma linha na imagem. Se, em vez disso, o robô é equipado com um sensor a laser 2D, esse vetor pode representar a posição de um canto ou uma linha no quadro do sensor a laser.

De acordo com terminologia definida, podemos agora definir o SLAM como o problema de recuperar um modelo do mapa M e a trajetória do robô X_T a partir da odometria U_T e as observações Z_T . Na literatura, distinguimos entre o problema completo do SLAM e o problema do SLAM online. O problema de SLAM completo consiste em estimar a probabilidade posterior da articulação através de X_T e M dos dados, isto é:

$$p(X_T, M | Z_T, U_T). \quad (2.18)$$

O problema de SLAM online, por sua vez, consiste em estimar a probabilidade posterior da articulação através de x_T e M dos dados, logo:

$$p(x_T, M | Z_T, U_T). \quad (2.19)$$

Portanto, o problema de SLAM completo tenta recuperar toda a trajetória do robô X_T , enquanto o problema de SLAM online tenta estimar apenas a pose atual do robô x_T .

2.3.2 EKF-SLAM

O EKF-SLAM é historicamente a primeira formulação proposta e procede exatamente como o EKF padrão usado para localização, com a única diferença de usar um vetor de estado estendido y_t que compreende tanto a posição do robô x_t e a posição de todas as características m_i no mapa, ou seja:

$$y_t = [x_t, m_0, \dots, m_{n-1}]^T \quad (2.20)$$

À medida que novas características são observadas, elas são adicionadas ao vetor de estado. Deste modo, a matriz de covariância do ruído cresce de forma quadrática. Por razões computacionais, o tamanho do mapa é, portanto, geralmente limitado a menos de mil características. No entanto, várias abordagens foram desenvolvidas para lidar com um número maior de recursos, que decompõem o mapa em mapas menores, para os quais as covariâncias são atualizadas separadamente (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

O EKF-SLAM foi aplicado com sucesso em diversos domínios, incluindo ambientes aéreos, subaquáticos, externos e internos. A Figura 4 ilustra o princípio de funcionamento do EKF-SLAM em um ambiente simples com três características. Neste exemplo, a localização inicial do robô é assumida como a origem do sistema de referências e, portanto, a incerteza inicial da pose do robô é zero. A partir dessa posição, o robô observa uma característica e a mapeia com uma incerteza relacionada ao modelo de erro do sensor (a). À medida que o robô se move, sua incerteza da pose aumenta pelo efeito dos erros introduzidos pela odometria (b). Em seguida, o robô observa mais duas características e as mapeia com uma incerteza que resulta da combinação do erro de medição com a incerteza da posição do robô (c). A partir disso, nota-se que o mapa se correlaciona com a estimativa da posição do robô. Agora, o robô dirige de volta para sua posição inicial e sua incerteza aumenta novamente (d). Neste ponto, ele observa a primeira característica, cuja localização é relativamente bem conhecida em comparação com os outros recursos. Isso torna o robô mais seguro sobre sua localização atual e, portanto, sua incerteza de pose diminui (e). Observe que até agora só consideramos o problema do SLAM online. Portanto, somente a posição atual do robô estava sendo atualizada. O problema completo do SLAM, por outro lado, atualiza todo o percurso do robô e, portanto, todas as poses anteriores. Nesse caso, depois de observar novamente a primeira característica, a incerteza da posição anterior do robô dimi-

nuirá e também as incertezas associadas aos outras características. A posição dessas características está de fato correlacionada com as poses anteriores do robô (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

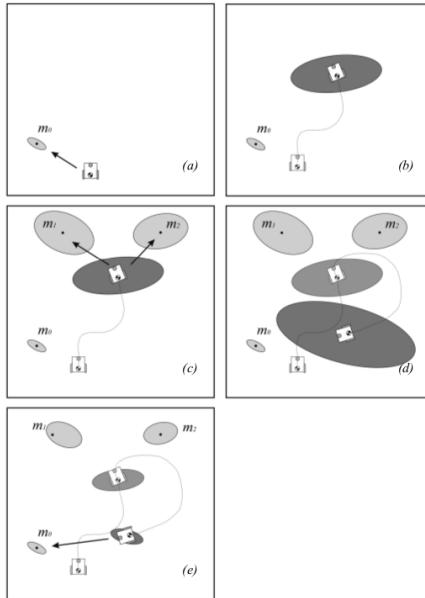


Figura 4 – Ilustração de um exemplo EKF-SLAM (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

Outra desvantagem em relação ao EKF-SLAM está na linearização feita pelo filtro de Kalman estendido, que é refletida pelo uso de Jacobianas nas atualizações de movimento e medição. Infelizmente, ambos os modelos de movimento e medição são tipicamente não-lineares, portanto suas linearizações podem, às vezes, levar à inconsistência ou divergência da solução. Além disso, outro problema no EKF-SLAM é sua sensibilidade a associações incorretas de dados das características, o que acontece quando o robô combina incorretamente ao recurso m_i com recursos m_j . Esse problema se torna ainda mais importante no fechamento de ciclo (*loop closure*), ou seja, quando o robô retorna para reobservar características após uma longa travessia. A associação incorreta de dados pode ocorrer com frequência com os sensores a laser 2D devido à dificuldade de identificar características distintas

em suas nuvens de pontos. Este problema pode ser facilitado com câmeras, graças à enorme disponibilidade de detectores de características (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

2.3.3 SLAM baseado em grafos

Introduzido por Lu e Milios (1997), o SLAM baseado em grafos nasceu da intuição de que o problema do SLAM pode ser interpretado como um grafo esparsa de nós e restrições entre nós. Os nós do grafo são as poses do robô (x_0, x_1, \dots, x_T) e as n características no mapa (m_0, m_1, \dots, m_{n-1}). As restrições são a posição relativa entre poses consecutivas do robô e a posição relativa entre a poses do robô e as características observadas a partir desses locais (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

A Figura 5 ilustra um exemplo de construção de grafo em um cenário simples com um robô diferencial. De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), a principal propriedade do SLAM baseado em grafos é que as restrições não devem ser consideradas como restrições rígidas, mas como restrições suaves, pois é relaxando essas restrições que podemos calcular a solução para o problema completo do SLAM, ou seja, a melhor estimativa do trajeto do robô e do mapa do ambiente. Em outras palavras, o SLAM baseado em grafos representa as localizações do robô e características como os nós de uma rede elástica. A solução SLAM pode ser encontrada computando o estado de energia mínima desta rede.

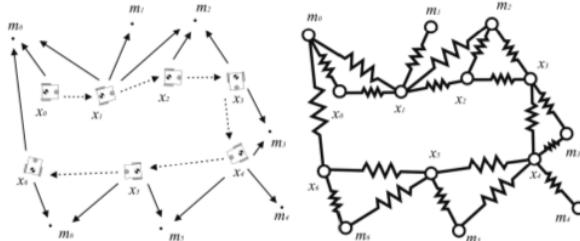


Figura 5 – Exemplo de construção de grafo. Restrições entre nós no grafo são representadas com restrições suaves (como molas) (SIEGWART; NOURBAKSH; SCARAMUZZA, 2011).

Existe uma vantagem significativa das técnicas de SLAM base-

adas em grafos sobre o EKF-SLAM. No EKF SLAM, o custo computacional e o requisito de memória para atualizar e armazenar a matriz de covariância cresce de maneira quadrática com o número de características. Por outro lado, no SLAM baseado em grafos, o tempo de atualização do grafo é constante e a memória necessária é linear com o número de características. No entanto, a otimização final do gráfico pode se tornar computacionalmente onerosa se a trajetória do robô for longa. Existem algoritmos SLAM baseados em grafos mostraram resultados impressionantes e muito bem-sucedidos com até cem milhões de recursos. No entanto, esses algoritmos tentam otimizar todo o caminho do robô e, portanto, foram implementados para trabalhar offline. Algumas das implementações online usaram abordagens de subdivisão do mapa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.3.4 SLAM com filtro de partículas (FastSLAM)

Essa solução específica para o problema SLAM é baseada na amostragem aleatória da distribuição. O termo filtro de partículas nasce do fato de que ele não representa a distribuição de crenças do robô em uma forma paramétrica (como Gaussiana), mas sim como um conjunto de amostras (isto é, partículas) extraídas aleatoriamente desta distribuição. O poder dessa representação está em sua capacidade de modelar qualquer tipo de distribuição (por exemplo, não-Gaussiana), além de transformações não-lineares (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Os filtros de partículas encontram sua origem nos métodos de Monte Carlo (METROPOLIS; ULAM, 1949), mas um passo os tornaram praticamente aplicáveis ao problema de SLAM é baseado no trabalho de Rao (1992) e Blackwell (1947), do qual esses filtros herdaram o nome de filtros de partículas *Rao-Blackwellized*. Finalmente, o filtro de partículas *Rao-Blackwellized* foi aplicado pela primeira vez ao problema de SLAM por Murphy e Russell (2001) e uma implementação muito eficiente é encontrada no trabalho de Montemerlo et al. (2002), que também usou o nome de FastSLAM.

Numa visão geral, a cada passo de tempo, o filtro de partículas mantém sempre o mesmo número K de partículas (por exemplo, $K = 1000$). Cada partícula contém uma estimativa do trajeto do robô $X_t^{[K]}$ e estimativas da posição de cada característica no mapa, que são representadas por Gaussianas bidimensionais com valores médios $\mu_{t,i}^{[K]}$ e matrizes de covariância $\Sigma_{t,i}^{[K]}$. Portanto, uma partícula é caracterizada

por:

$$X_t^{[K]}; (\mu_{t,0}^{[K]}, \Sigma_{t,0}^{[K]}); (\mu_{t,1}^{[K]}, \Sigma_{t,1}^{[K]}); \dots; (\mu_{t,n-1}^{[K]}, \Sigma_{t,n-1}^{[K]}), \quad (2.21)$$

onde k denota o índice da partícula e n o número de características no mapa. No SLAM por filtro de partículas, a média e covariância de cada característica é atualizada usando um filtro de Kalman diferente para cada característica (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Quando o robô se move, o modelo de movimento especificado pela leitura odométrica u_t é aplicado em cada partícula $x_{t-1}^{[K]}$ para gerar o novo local $x_t^{[K]}$. Quando o robô faz uma observação z_t , é calculada para cada partícula o chamado fator de importância $w_t^{[K]}$, que é determinado como a probabilidade de observar z_t , dado a partícula $x_t^{[K]}$ e todas as observações anteriores $z_{0 \rightarrow t-1}$:

$$w_t^{[K]} = p(z_t | x_t, z_{0 \rightarrow t-1}) \quad (2.22)$$

A etapa final deste processo é chamada reamostragem, na qual é substituído o conjunto atual de partículas por outro conjunto de acordo com o fator de importância determinado anteriormente. Por fim, a média e covariância de cada característica são atualizadas de acordo com a regra de atualização EKF padrão (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), Embora possa parecer complexo, o algoritmo FastSLAM pode ser prontamente implementado e é um dos algoritmos SLAM mais fáceis de implementar. Além disso, o FastSLAM tem a grande vantagem sobre o EKF-SLAM de sua complexidade crescer de maneira logarítmica com o número de características (não quadraticamente como no EKF-SLAM). Isso ocorre principalmente porque, em vez de usar uma única matriz de covariância para a pose do robô e o mapa (como no EKF-SLAM), ele usa filtros Kalman separados, um para cada característica.

Além disto, outra grande vantagem sobre o EKF-SLAM é que, devido ao uso de amostragem randomizada, ele não requer a linearização do modelo de movimento e também pode representar distribuições não-gaussianas.

2.4 CONCEITOS RELACIONADOS À VISÃO COMPUTACIONAL

A utilização de visão computacional para navegação, especialmente para as tarefas de localização e mapeamento, tem sido alvo de intenso estudo por parte de pesquisadores (DESOUZA; KAK, 2002) (BONIN-FONT; ORTIZ; OLIVER, 2008).

Em geral, o problema de localização baseado em imagens consiste na detecção e descrição de características locais em uma imagem e a comparação entre essas características e suas semelhantes em imagens anteriores ou imagens chave guardadas em um banco de dados. As principais técnicas conhecidas na literatura são apresentadas nesta seção.

2.4.1 Detectores e descritores de características locais

Descritores de características locais são uma solução para o problema de encontrar marcações naturais em um ambiente. Uma característica local é basicamente um padrão de imagem que difere da sua vizinhança imediata. Os detectores devem fornecer regiões que são utilizadas para calcular os descritores, extraíndo as características da imagem. Uma vez que as características são detectadas, elas podem ser extraídas. O extrator calcula um descritor dos pixels na região em torno de cada ponto de interesse (TUYTELAARS; MIKOLAJCZYK et al., 2008).

Segundo Mikolajczyk e Schmid (2005), diversas técnicas para descrever regiões de interesse em imagem têm sido desenvolvidas. O descritor mais simples é um vetor de pixels da imagem. Correlação cruzada pode então ser empregada para calcular um grau de similaridade entre dois descritores. No entanto, a alta dimensionalidade de tal descrição resulta em uma alta complexidade computacional para o reconhecimento. Portanto, esta técnica é usada principalmente para encontrar correspondências entre duas imagens.

Aqui são apresentados alguns detectores e descritores conhecidos e de ampla utilização em visão computacional.

2.4.1.1 SIFT

O descritor SIFT (*Scale-invariant Feature Transform*) transforma uma imagem em uma coleção de vetores de características locais, sendo

tolerante a translação, rotação e escala de imagens, e parcialmente invariante a mudanças de iluminação e projeções afim ou 3D (LOWE, 1999). A primeira etapa da detecção de ponto de interesse (*keypoint*) é identificar locais e escalas que podem ser repetidamente atribuídos sob diferentes visões a um mesmo objeto (LOWE, 1999).

De acordo com Lowe (2004), o espaço-escala de uma imagem é definido como uma função, $L(x,y,\sigma)$, que é produzida a partir da convolução de uma escala variável Gaussiana, $G(x,y,\sigma)$, com uma imagem de entrada, $I(x,y)$:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y), \quad (2.23)$$

onde σ é o fator de escala, $*$ é a operação de convolução em x e y, e $G(x,y,\sigma)$ é definida por:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.24)$$

Para detectar eficientemente localizações de pontos de interesse estáveis em escala-espaço, Lowe (1999) utiliza extremos de escala-espaço na função Diferença-entre-Gaussianas (DeG) convolvida com a imagem, $D(x,y,\sigma)$, calculada a partir da diferença de duas escalas próximas separadas por um fator multiplicativo constante k:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \quad (2.25)$$

A fim de detectar os máximos locais e mínimos de $D(x,y,\sigma)$, cada ponto amostrado é comparado com seus oito vizinhos na imagem atual e nove vizinhos na escala acima e abaixo. Esse ponto é selecionado somente se ele é maior ou menor do que todos esses vizinhos (LOWE, 1999). Assim, os *keypoints* são um subconjunto dos máximos e mínimos da diferença entre as escalas da pirâmide de imagens.

Após a obtenção dos pontos de máximo e mínimo, a definição dos pontos de interesse é feita através de um processo de filtragem em nível de sub-pixel. Para isso, Lowe (1999) utiliza uma expansão de Taylor de segunda ordem da função de escala-espaço, $D(x,y,\sigma)$, em torno do ponto de máxima ou mínima.

As DeG têm alta resposta para as bordas, que também precisam ser removidas. Para isso, um conceito semelhante ao detector de cantos de Harris é usado. Uma matriz 2 x 2 Hessiana (H) é usada para calcular a curvatura principal. Para o detector de cantos de Harris, um valor próprio é maior que o outro para as bordas. Lowe (1999) usa uma

função simples: se esta proporção é superior a um valor limiar, esse ponto de interesse é descartado.

A escala do ponto de interesse é usada para selecionar a imagem suavizada pela Gaussiana, L , com a escala mais próxima, de modo que todos os cálculos são efetuados em escala invariante. Para cada amostra de imagem, $L(x,y)$, a esta escala, a magnitude do gradiente, $m(x,y)$, e a orientação, $\theta(x,y)$, são pré-calculados usando diferenças de pixels (LOWE, 1999):

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.26)$$

$$\theta(x,y) = \tan^{-1} \left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \right) \quad (2.27)$$

Em cada ponto de interesse obtido, um descritor de imagem é calculado, que pode ser visto como um histograma dependente da posição das direções do gradiente local na vizinhança do ponto de interesse (SANDE; GEVERS; SNOEK, 2010). Cada ponto é utilizado para gerar um vetor de características que descreve a região da imagem amostrada relativa à sua coordenada escala-espacão da imagem. As características atingem invariância parcial a variações locais, como projeções afim ou 3D, desfocando gradientes locais da imagem (LOWE, 1999). São definidas $n \times n$ regiões com $k \times k$ pixels ao redor do ponto de interesse detectado, geralmente com $n = 2$ ou 4 e $k = 4$. Para cada região, é feito um histograma em 8 direções com as magnitudes dos pixels. Assim, um total de 128 valores estão disponíveis. São representados como um vetor para formar o descritor do ponto de interesse (KAMILA, 2015).

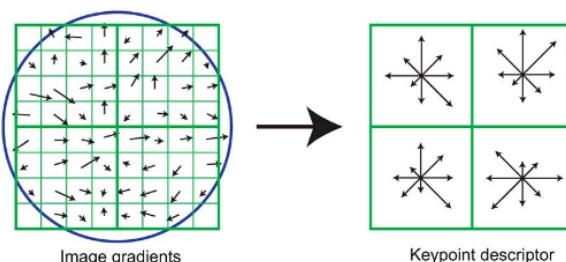


Figura 6 – Descritor SIFT (LOWE, 2004)

2.4.1.2 SURF

O descriptor SURF (*Speed-Up Robust Features*) é baseado em propriedades semelhantes às do SIFT, com uma complexidade reduzida. Em SIFT, o Laplaciano do Gaussiano (LoG) é aproximado com DeG para encontrar o espaço-escala. SURF utiliza *box filter* para realizar essa aproximação (BAY; TUYTELAARS; GOOL, 2006).

De acordo com Bay, Tuytelaars e Gool (2006), uma vantagem desta aproximação é que a convolução com *box filter* pode ser facilmente calculada com a ajuda de imagens integrais. E isso pode ser feito em paralelo para diferentes escalas. Bay, Tuytelaars e Gool (2006) baseiam seu detector na matriz Hessiana, justificando por seu bom desempenho em tempo de computação e precisão. No entanto, em vez de usar uma medida diferente para selecionar o local e a escala, SURF utiliza o determinante da Hessiana para ambos.

Dado um ponto $p = (x, y)$ em uma imagem I , a matriz Hessiana $H(p, \sigma)$ em p na escala σ é definida por (BAY; TUYTELAARS; GOOL, 2006):

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (2.28)$$

Onde $L_{xx}(x, \sigma)$ é a convolução da derivada gaussiana de segunda ordem $\frac{\partial^2}{\partial x^2} g(\sigma)$ com a imagem I no ponto p , e similarmente para $L_{xy}(x, \sigma)$ e $L_{yy}(x, \sigma)$.

Espaços-escala são geralmente implementadas como pirâmides de imagem. As imagens são repetidamente suavizadas com uma Gaussiana e posteriormente sub-amostradas, a fim de atingir um nível mais elevado da pirâmide. Devido ao uso de *box filter* e imagens integrais, Bay, Tuytelaars e Gool (2006) afirmam que não é necessário aplicar iterativamente o mesmo filtro para a saída de uma camada filtrada anteriormente, mas em vez disso pode aplicar tais filtros de qualquer tamanho exatamente à mesma velocidade diretamente sobre a imagem original. Portanto, o espaço-escala é analisado por aumento de resolução do tamanho do filtro em vez de reduzir de forma iterativa o tamanho da imagem. A saída do *box filter* 9x9 é considerada como a camada de escala inicial, $s = 1, 2$ (correspondente a derivadas gaussianas com $\sigma = 1, 2$). As camadas seguintes são obtidas filtrando a imagem com máscaras gradualmente maiores, levando em consideração a natureza discreta das imagens integrais e a estrutura específica dos filtros. Em escalas maiores, o passo entre os tamanhos dos filtros consecutivos

também deve ser dimensionado em conformidade. Assim, o tamanho do filtro é dobrado para cada nova oitava. Simultaneamente, os intervalos de amostragem para a extração dos pontos de interesse podem ser duplicados (BAY; TUYTELAARS; GOOL, 2006).

Segundo Bay, Tuytelaars e Gool (2006), com o intuito de localizar pontos de interesse na imagem e sobre escalas, uma supressão não máxima numa vizinhança $3 \times 3 \times 3$ é aplicada. Então, os máximos do determinante da matriz Hessiana são interpolados em escala e espaço da imagem.

A fim de ser invariante a rotação, Bay, Tuytelaars e Gool (2006) utilizam as respostas *Haar Wavelet* em direções x e y dentro de uma vizinhança circular de raio $6s$ ao redor do ponto de interesse. Uma vez que essas respostas são calculadas e ponderadas com um Gaussiano ($\sigma = 2,5s$) centradas no ponto de interesse, elas são representadas como vetores em um espaço bidimensional com a intensidade de resposta horizontal ao longo da abscissa e a de resposta vertical ao longo da ordenada. A orientação dominante é estimada calculando a soma de todas as respostas dentro de uma janela de orientação deslizante cobrindo um ângulo de $\frac{\pi}{3}$.

Bay, Tuytelaars e Gool (2006) argumentam que para detectar pontos de interesse, SURF usa uma aproximação inteira do determinante do detector hessiano de *blob* (regiões em uma imagem que diferem em propriedades, como brilho ou cor, em comparação com as regiões circundantes), que pode ser calculado com três operações inteiras usando uma imagem integral pré-computada. O descritor SURF é baseado na soma da resposta *Haar Wavelet* em torno do ponto de interesse, que também podem ser calculados com a ajuda da imagem integral.

A próxima etapa consiste em fixar uma orientação reproduzível com base em informação a partir de uma área circular em torno do ponto de interesse. Em seguida, uma região quadrada alinhada com a orientação selecionada é construída, e o descritor SURF pode ser extraído a partir dele. A região de interesse é dividida regularmente em 4×4 sub-regiões quadradas, e para cada uma, as respostas *Haar Wavelet* são extraídas em 5×5 pontos de amostragem regularmente espaçados. Elas são ponderadas com uma Gaussiana para oferecer maior robustez para deformações, ruídos e translações (BAY; TUYTELAARS; GOOL, 2006).

2.4.1.3 BRIEF

Binary Robust Independent Elementary Features (BRIEF), proposto por Calonder et al. (2010), é um descritor binário de uso geral robusto às classes típicas de transformações de imagem fotométricas e geométricas que pode ser combinado com detectores arbitrários.

O algoritmo SIFT usa um vetor de dimensão 128 para descritores e uma vez que utiliza números de ponto flutuante, requer aproximadamente 512 bytes. Do mesmo modo, SURF também consome um mínimo de 256 bytes (CALONDER et al., 2010). O problema é que quanto mais memória, maior é o tempo necessário para o processo de correspondência. Os descritores podem ser comprimidos usando vários métodos como PCA (Análise de Componentes Principais) (JOLLIFFE, 1986) e LDA (LI; YUAN, 2005). LSH (*Local Sensitive Hashing*) (KULIS; GRAUMAN, 2009) também é utilizado para converter esses descritores SIFT de números de ponto flutuante para cadeias binárias. Essas sequências representam descritores binários que são usados para combinar características usando a distância de Hamming (XOR seguido por uma contagem de bits) e podem substituir a habitual distância euclidiana (ALAHİ; ORTIZ; VANDERGHEYNST, 2012).

A redução de dimensionalidade requer primeiro calcular o descritor completo antes do processamento adicional. Computando diretamente cadeias binárias nas regiões da imagem, BRIEF não precisa deste processo. Os bits individuais são obtidos através da comparação das intensidades de pares sem a necessidade de uma fase de treinamento (CALONDER et al., 2010).

Calonder et al. (2010) cria um vetor com as respostas dos testes, que são calculados após a suavização da região da imagem. No caso, define-se um t teste na região p de tamanho $S \times S$ como:

$$t(p; x, y) := \begin{cases} 1, & \text{se } p(x) < p(y) \\ 0, & \text{caso contrário} \end{cases} \quad (2.29)$$

Onde $p(x)$ é a intensidade do pixel após a suavização. A escolha exclusiva de um conjunto de pares de localização $n_d(x, y)$ define um conjunto de testes binários (CALONDER et al., 2010).

$$f_{nd}(p) = \sum_{1 \leq i \leq nd} 2^{i-1} t(p; x_i, y_i) \quad (2.30)$$

O descritor BRIEF utiliza uma região de imagem suavizada e seleciona um conjunto de pares (x, y) . Em seguida, são feitas com-

parações de intensidade de pixel nesses locais, resultando em 1 ou 0. De acordo com Calonder et al. (2010), experimentos mostraram que somente 256 bits, ou até mesmo 128 bits, muitas vezes são suficientes para obter resultados muito bons de correspondência. A escolha de um conjunto de pares (x, y) exclusivos define um conjunto de testes binários. O descritor obtido não é invariante a mudanças de escala e rotação, a menos que utilizado juntamente com um detector capaz de prover tal habilidade.

2.4.1.4 ORB

O algoritmo ORB (*Oriented FAST and Rotated BRIEF*) (RUBLEE et al., 2011) é uma combinação do detector FAST (ROSTEN; DRUMMOND, 2006) com o descritor BRIEF e algumas modificações para melhorar o desempenho.

Na fase de detecção de *keypoints*, Rublee et al. (2011) aplica o algoritmo FAST para encontrar pontos de interesse, em seguida, utiliza o detector de cantos de Harris para encontrar os melhores N pontos entre eles. ORB também usa pirâmide para produzir características multiescalas, pois FAST não as produz. Um problema é que FAST não calcula a orientação. Para resolver este problema, Rublee et al. (2011) utiliza uma medida de orientação de canto, a intensidade do centroide, definida por Rosin (1999) que assume que a intensidade de um canto está deslocada do seu centro, e este vetor pode ser utilizado para gerar uma orientação. Rosin (1999) define o momento da região como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \quad (2.31)$$

Com isso, pode-se encontrar o centroide:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.32)$$

A orientação da região é obtida por:

$$\theta = \text{atan2}(m_{01}, m_{10}) \quad (2.33)$$

onde *atan2* retorna ângulo cuja tangente é o quociente de dois números especificados respeitando o quadrante.

Para melhorar a invariância a rotação desta medida, os momentos são calculados com x e y permanecendo dentro de uma região cir-

cular de raio r (RUBLEE et al., 2011).

Em relação aos descritores, ORB utiliza BRIEF. Para resolver o mal desempenho de BRIEF com rotação, ORB utiliza a versão orientada de BRIEF (oBRIEF), rotacionando a direção do descritor de acordo com a orientação do *keypoint* (RUBLEE et al., 2011).

Rublee et al. (2011) afirma que qualquer conjunto de características de n testes binários no local (x_i, y_i) , define a matriz 2 x n:

$$S = \begin{pmatrix} x_1, & \dots, & x_n \\ y_1, & \dots, & y_n \end{pmatrix} \quad (2.34)$$

Usando a orientação θ da região e a matriz de rotação correspondente R_θ , Rublee et al. (2011) constrói uma versão rotacionada S_θ de S:

$$S_\theta = R_\theta S \quad (2.35)$$

Então, a versão rotacionada de BRIEF é dada por:

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S_\theta \quad (2.36)$$

Assim, Rublee et al. (2011) discretiza o ângulo em incrementos de 12 graus, e constrói uma tabela de pesquisa de padrões BRIEF pré-computados. Enquanto a orientação θ do ponto de interesse é consistente em todos os pontos de vista, o conjunto correto de pontos S_θ será usado para calcular o seu descritor.

Ao contrário do descritor BRIEF, ORB é relativamente invariante a escala e rotação enquanto ainda emprega a eficiente distância de Hamming no processo de combinação (RUBLEE et al., 2011).

2.4.1.5 BRISK

A detecção de *keypoints* de BRISK (*Binary Robust Invariant Scalable Keypoints*) é inspirada no detector AGAST (MAIR et al., 2010), que, por sua vez, é uma extensão do detector FAST para desempenho acelerado (LEUTENEGGER; CHLI; SIEGWART, 2011).

As camadas da pirâmide escala-espacó consistem, tipicamente de 4 oitavas e 4 intra-oitavas. As oitavas são formados progressivamente por sub-amostragens (metade) da imagem original. A primeira intra-oitava é obtida pela sub-amostragem da imagem original por um fator de 1,5, enquanto o resto das camadas intra-oitavas são derivadas por sub-amostragens sucessivas. Deste modo, cada intra-oitava está locali-

zada entre camadas de oitavas (LEUTENEGGER; CHLI; SIEGWART, 2011).

Inicialmente, Leutenegger, Chli e Siegwart (2011) utilizam o detector FAST 9-16 em cada oitava e intra-oitava separadamente usando o mesmo limiar para identificar potenciais regiões de interesse. Em seguida, os pontos pertencentes a estas regiões são submetidos a uma supressão não-máxima em escala-espacó: o ponto em questão tem de cumprir a condição de máxima pontuação FAST em relação aos seus oito vizinhos na mesma camada e nas camadas acima e abaixo. A pontuação FAST é definida como o limite máximo ainda considerando um ponto de imagem como um canto.

Dado um conjunto de *keypoints* (composto por localizações refinadas de sub-pixéis e valores associados da escala), o descritor BRISK é composto por uma cadeia binária que concatena os resultados de testes de comparação de brilho simples (LEUTENEGGER; CHLI; SIEGWART, 2011).

De acordo com Leutenegger, Chli e Siegwart (2011), o descritor BRISK faz uso de um padrão, ilustrado na Figura 7, que define locais igualmente espaçados em círculos concêntricos com o ponto de interesse. Para evitar os efeitos de *aliasing* no momento da amostragem, a intensidade da imagem de um ponto no padrão, BRISK utiliza suavização Gaussiana com desvio padrão proporcional à distância entre os pontos sobre o respectivo círculo.

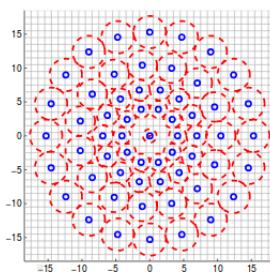


Figura 7 – O padrão de amostragem BRISK com $N = 60$ pontos: os pequenos círculos azuis indicam os locais de amostragem; os maiores, círculos vermelhos tracejados, são desenhados num raio σ correspondente ao desvio padrão do núcleo Gaussiano utilizado para suavizar os valores de intensidade nos pontos de amostragem (LEUTENEGGER; CHLI; SIEGWART, 2011).

Segundo Leutenegger, Chli e Siegwart (2011), os valores de intensidade suavizados nesses pontos são utilizados para estimar o gradiente local por:

$$g(p_i, p_j) = (p_i - p_j) \frac{I(p_j, \sigma_j) - I(p_i, \sigma_{ij})}{\|p_j - p_i\|^2} \quad (2.37)$$

Considerando o conjunto de todos os pares de pontos de amostragem, Leutenegger, Chli e Siegwart (2011) definem um subconjunto de pareamentos de curta distância (S) e outro subconjunto de longa distância (L) de acordo com a escala do ponto de interesse. A distância limiar é definida proporcionalmente a escala do *keypoint*. Iterando através dos pares de pontos em \mathcal{L} , a estimativa da direção do ponto de interesse k é feita por:

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{p_i, p_j \in \mathcal{L}} g(p_i, p_j) \quad (2.38)$$

Para a formação do descritor com rotação e escala normalizados, Leutenegger, Chli e Siegwart (2011) utilizam o padrão de amostragem rotacionado por $\alpha = \arctan 2(g_y, g_x)$ em torno do ponto de interesse k. O descritor é um vetor de bits montado através da realização de todas as comparações de intensidade nos pares de ponto (p_j^α, p_i^α) de S.

$$b = \begin{cases} 1, & \text{se } I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, & \text{caso contrário} \end{cases} \quad (2.39)$$

Um exemplo de combinação entre imagens utilizando BRISK pode ser observado na Figura 8.

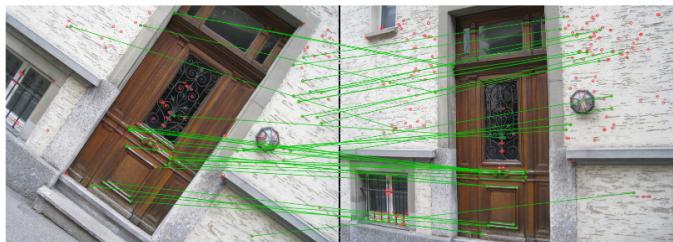


Figura 8 – Exemplo de combinação com BRISK. As combinações resultantes são conectadas por linhas verdes, que não mostram falsos positivos claros (LEUTENEGGER; CHLI; SIEGWART, 2011).

2.4.1.6 FREAK

Fast Retina Keypoint (FREAK), proposto por Alahi, Ortiz e Vandergheynst (2012), é um descritor de pontos de interesse inspirado no sistema visual humano, mais especificamente na retina. Uma cascata de cadeias binárias é calculada comparando intensidades de imagem ao longo de um padrão de amostragem semelhante ao da retina.

Muitas redes de amostragem são possíveis para comparar pares de intensidades de pixels. Alahi, Ortiz e Vandergheynst (2012) utilizam uma malha de amostragem inspirada na retina humana que é circular, assim como BRISK, com a diferença de ter maior densidade de pontos perto do centro. A densidade de pontos cai exponencialmente como pode ser visto na Figura 9. Cada ponto de amostragem deve ser suavizado para tornar-se menos sensível ao ruído.

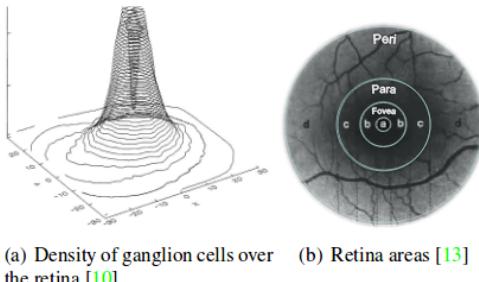


Figura 9 – (a) Densidade de células ganglionares sobre a retina (b) Áreas da retina (ALAHÍ; ORTIZ; VANDERGHEYNST, 2012).

Para coincidir com o modelo de retina, Alahi, Ortiz e Vandergheynst (2012) utilizam núcleos de tamanhos diferentes para cada ponto amostral, assim como BRISK, porém com uma mudança exponencial no tamanho dos campos receptivos sobrepostos. A Figura 10 ilustra a topologia dos campos receptivos. Cada círculo representa os desvios-padrão dos núcleos Gaussianos aplicados aos pontos de amostragem correspondentes. Esses núcleos têm tamanho variável com respeito ao log-polar do padrão da retina.

O descritor binário é construído através da limitação da diferença entre pares de campos receptivos com o seu núcleo Gaussiano correspondente (ALAHÍ; ORTIZ; VANDERGHEYNST, 2012). Em outras palavras, o descritor é uma cadeia binária formada por uma sequência

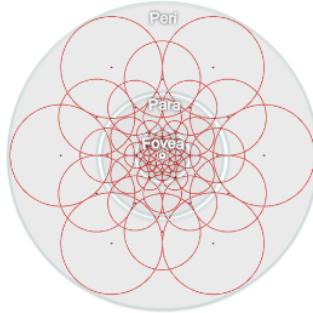


Figura 10 – Ilustração do padrão de amostragem FREAK em que cada círculo representa um campo receptivo onde a imagem é suavizada com o respectivo núcleo Gaussiano (ALAH; ORTIZ; VANDERGHEYNST, 2012).

de diferença entre Gaussianas de 1 bit:

$$F = \sum_{0 \leq \alpha \leq N} 2^\alpha T(P_\alpha), \quad (2.40)$$

Onde P_α é um par de campos receptivos, N é o tamanho desejado do descriptor, e

$$T(P_\alpha) = \begin{cases} 1, & \text{se } I(p_\alpha^{r1}) - I(p_\alpha^{r2}) > 0 \\ 0, & \text{caso contrário} \end{cases} \quad (2.41)$$

Sendo $I(p_\alpha^{r1})$ a intensidade suavizada do primeiro campo receptivo do par p_α .

Com algumas dezenas de campos receptivos, milhares de pares são possíveis o que ocasiona em um grande descriptor. No entanto, muitos dos pares podem não ser úteis para descrever de forma eficiente uma imagem. Para selecionar os pares, FREAK inicialmente cria uma matriz com cerca de cinquenta mil *keypoints* extraídos. Cada linha corresponde a um ponto de interesse representado por seu descriptor composto de todos os pares possíveis no padrão de amostragem da retina. Em seguida, calcula-se a média de cada coluna. De modo a ter uma característica discriminante, uma alta variância é desejada, portanto as colunas são ordenadas de acordo com a maior variância, armazenando a melhor coluna (maior variância) e iterativamente adicionando colunas remanescentes que possuam baixa correlação com as colunas

selecionadas (ALAHI; ORTIZ; VANDERGHEYNST, 2012).

Para estimar a rotação do *keypoint*, Alahi, Ortiz e Vanderghenst (2012) somam os gradientes locais dos pares selecionados similarmente ao BRISK, porém selecionam pares com campos receptivos simétricos em relação ao centro. FREAK seleciona 45 pares em oposição às centenas de pares de BRISK. Além disso, o padrão de retina tem campos receptivos maiores na região perifoveal do que BRISK permitindo um maior erro na estimativa de orientação.

2.4.1.7 GFTT

Introduzido por Shi e Tomasi (1993), o detector *Good Features to track* (GFTT) é um método para seleção de características, um algoritmo de rastreamento baseado em um modelo de mudanças de imagens que mantém semelhanças entre si e uma técnica para monitorar as características durante o processo de rastreamento.

A seleção maximiza especificamente a qualidade do rastreamento e, portanto, é ideal por construção, em oposição a demais medidas de texturização para esta finalidade. A monitoração é computacionalmente barata, sólida e ajuda a discriminar entre características boas e ruins com base em uma medida de dissimilaridade que usa movimento afim (*Affine Motion*) como o modelo de mudança de imagem subjacente.

2.4.2 RANSAC

O algoritmo *Random Sample Consensus* (RANSAC), proposto por Fischler e Bolles (1987), propõe uma abordagem de estimação de parâmetros gerais projetada para lidar com uma grande proporção de *outliers* nos dados de entrada. Ao contrário de muitas das técnicas de estimação robustas comuns que foram adotadas pela comunidade de visão computacional a partir da literatura estatística, RANSAC foi desenvolvido pela comunidade de visão computacional, sendo amplamente utilizado para detectar o problema de falsas correspondências. A porcentagem de *outliers* que podem ser manipulados pelo RANSAC pode ser maior do que 50% de todo o conjunto de dados (ZULIANI, 2009).

Uma suposição básica é que os dados são compostos por *inliers*, isto é, dados cuja distribuição pode ser explicada por um conjunto de

parâmetros do modelo, embora possam estar sujeitos a ruído, e *outliers*, que são dados que não se encaixam no modelo. Zuliani (2009) afirma que o dado é considerado um *outlier* se ele não atender o modelo, considerado verdadeiro, instanciado pelo conjunto de parâmetros, considerados verdadeiros, dentro de um limite de erro que define o desvio máximo atribuível ao efeito do ruído.

Apesar de muitas modificações terem sido propostas, o algoritmo RANSAC é essencialmente composto por duas etapas que são repetidas de uma forma iterativa: a etapa de hipótese, em que os primeiros conjuntos de amostras mínimas (MSSs) são selecionados aleatoriamente a partir do conjunto de dados de entrada e os parâmetros do modelo são calculados utilizando apenas os elementos do MSS; e a etapa de teste, em que o RANSAC verifica quais elementos do conjunto de dados são consistentes com o modelo instanciado com os parâmetros estimados a partir do primeiro passo. O conjunto de tais elementos é chamado conjunto de consenso (CS). O algoritmo termina quando a probabilidade de encontrar um CS melhor classificado cai abaixo de um certo limite (ZULIANI, 2009) (FISCHLER; BOLLES, 1987).

A Figura 11 apresenta um exemplo de aplicação simples do algoritmo RANSAC em um conjunto bidimensional de dados.

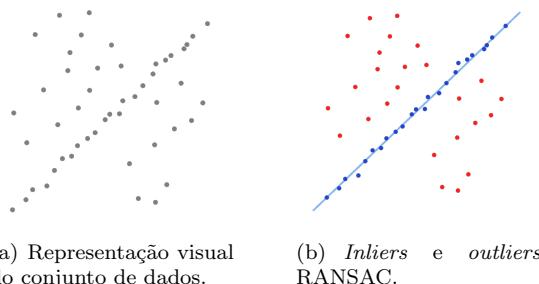


Figura 11 – Aplicação simples do algoritmo RANSAC em um conjunto bidimensional de dados. Gerado pelo Matlab.

2.4.3 Registro de imagens

O problema de consistentemente alinhar várias visualizações de imagens (duas ou mais) da mesma cena tomadas em momentos diferentes, de diferentes pontos de vista e/ou por sensores diferentes em um único modelo completo é conhecido como registro ou *registration*.

Seu objetivo é encontrar as posições e orientações relativas das visões adquiridas separadamente em uma estrutura de coordenadas globais, de forma que as áreas de intersecção entre elas se sobreponham perfeitamente. Um dos métodos mais populares é o algoritmo *Iterative Closest Point* (ICP) (ZHANG, 1994) (SHARP; LEE; WEHE, 2002), um método iterativo que tenta encontrar a transformação ideal entre dois conjuntos de dados minimizando média de erros de distâncias. O ICP usa pares de pontos 3D mais próximos na fonte e o modelo como correspondências e assume que cada ponto tem uma correspondência (RUSU et al., 2008) (ZITOVA; FLUSSER, 2003).

De acordo com Zitova e Flusser (2003), em geral, as abordagens do problema de registro podem ser divididas em quatro grupos principais de acordo com a maneira da aquisição de imagem:

- Pontos de vista diferentes (análise de múltiplas vistas): As imagens da mesma cena são adquiridas de diferentes pontos de vista. O objetivo é obter uma visão 2D maior ou uma representação 3D da cena digitalizada;
- Tempos diferentes (análise de múltiplos tempos): As imagens da mesma cena são adquiridas em espaços de tempo diferentes, frequentemente em base regular e, possivelmente, sob circunstâncias diferentes. O objetivo é encontrar e avaliar mudanças na cena que ocorreram entre as aquisições consecutivas de imagens;
- Sensores diferentes (análise multimodal): As imagens da mesma cena são adquiridas por sensores diferentes. O objetivo é integrar as informações obtidas de diferentes fontes para obter uma representação da cena mais complexa e detalhada;
- Cena para o modelo de reconstrução: Imagens de uma cena e um modelo da cena são registradas. O modelo pode ser uma representação computacional da cena ou outra cena com conteúdo similar. O objetivo é localizar a imagem adquirida na cena/modelo e/ou os comparar.

Segundo Basdogan e Oztireli (2008), as principais abordagens para resolver o problema de alinhamento de duas nuvens M e P podem se enquadrar em dois grandes grupos: uma baseia-se na minimização da distância entre M e P como no caso do amplamente utilizado algoritmo ICP, e a outra é baseada na seleção de características distintas comuns entre duas nuvens. A maioria dos métodos de registro baseados em seleção de características consiste nos seguintes quatro passos (ZITOVA; FLUSSER, 2003), conforme mostrado na Figura 12:

- Detecção de características: Objetos salientes e distintivos (regiões limítrofes, arestas, contornos, interseções de linhas, círculos, etc.) são detectados manualmente ou automaticamente, sendo preferidos métodos automáticos. Para processamento posterior, essas características podem ser representadas por seus representantes pontuais (centros de gravidade, terminações de linha, pontos distintivos, etc.), chamados pontos de controle (CPs) na literatura.
- Correspondência entre características: Neste passo, é estabelecida a correspondência entre as características detectadas na imagem atual e as detectadas na imagem de referência. Vários descritores de características e medidas de similaridade, em conjunto com relações espaciais entre as características são utilizados para essa finalidade.
- Estimativa do modelo de transformação: São estimados o tipo e os parâmetros das chamadas funções de mapeamento, alinhando a imagem detectada com a imagem de referência. Os parâmetros das funções de mapeamento são computados por meio da correspondência entre características utilizada.
- Reamostragem e transformação de imagens: A imagem detectada é transformada por meio das funções de mapeamento. Os valores de imagem em coordenadas não inteiras são calculados pela técnica de interpolação apropriada.

Apesar do processo de registro ser independente do alinhamento inicial das nuvens, geralmente os métodos baseados em características são mais lentos do que os em ICP (BASDOGAN; OZTIRELI, 2008).

Por definição, uma coleção de pontos 3D é referida como uma estrutura de nuvem de pontos (*point cloud*) (RUSU; COUSINS, 2011). Elas representam o formato básico de dados de entrada para sistemas de percepção 3D e fornecem representações discretas, mas significativas do mundo circundante.

A biblioteca de código aberto *Point Cloud Library* (PCL) apresenta uma abordagem avançada e abrangente para o tema da percepção 3D, e é destinada a fornecer suporte para todos os blocos de construção em 3D comuns que as aplicações precisam. A biblioteca contém algoritmos para: filtragem, estimativa de característica, reconstrução de superfície, registro, montagem de modelo e segmentação (RUSU; COUSINS, 2011).

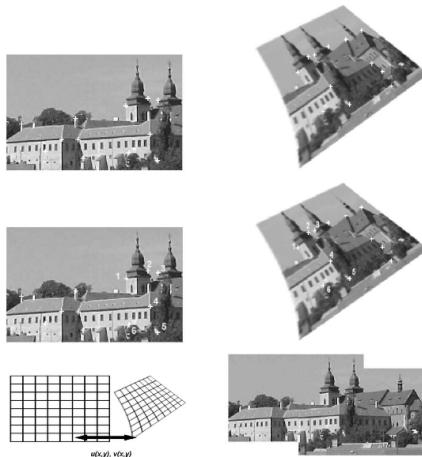


Figura 12 – Etapas de um processo de registro (neste caso, cantos foram utilizados como características (ZITOVA; FLUSSER, 2003).

Existem diversos sensores capazes de gerar essas nuvens de pontos, por exemplo câmeras estéreo. Esses dispositivos medem um grande número de pontos na superfície de um objeto, e muitas vezes emitem a nuvem de pontos como um arquivo de dados. A nuvem de pontos representa o conjunto de pontos que o dispositivo mediu. Outras alternativas para obtenção destas nuvens são sensores LiDAR e câmeras RGB-D, como o Microsoft Kinect, o Asus Xtion, o Ensenso Stereo 3D e o DepthSense, que podem estar acopladas em uma variedade de dispositivos, incluindo robôs móveis.

O sensor Microsoft Kinect, desenvolvido pela *PrimeSense*, é um dispositivo periférico (projeto inicialmente para jogos eletrônicos) que além de fornecer uma imagem RGB, também fornece um mapa de profundidade. Portanto, para cada pixel visto pelo sensor, o dispositivo também mede a distância ao sensor de acordo com suas limitações de distância (HÖGMAN, 2012). Usando um sensor de profundidade, essas câmeras evitam a complexidade da computação visual correspondente para estimativa de profundidade em combinações de imagens estéreo. Assim, sensores deste tipo vêm ganhando destaque em pesquisas relacionadas a localização e mapeamento.

Com a posse dos valores de profundidade, é possível desenhar uma nuvem de pontos, onde cada ponto é representado de forma a conter a posição 3D em coordenadas globais, valores de intensidade e

distância, como exemplificado na Figura 13.

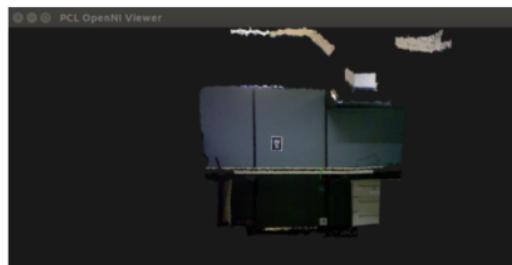


Figura 13 – Nuvem de pontos obtida pelo sensor Kinect com auxílio da biblioteca PCL (BARBOSA et al., 2017).

2.5 CONCLUSÃO

Esta seção apresentou uma série de conceitos fundamentais para o entendimento deste trabalho. Foram expostos conceitos que elucidam o problema de navegação robótica, especificamente nas tarefas de mapeamento e localização, e como esses problemas podem ser solucionados com técnicas de visão computacional, robótica probabilística e fusão de sensores.

Complementarmente, foram mostradas técnicas de localização e SLAM probabilísticos, bem como descritores de características locais, úteis no processo de encontrar marcações naturais, utilizadas nas fases de mapeamento e localização de robôs móveis. Por fim, foi mostrado o conceito de registro por nuvens de pontos que visa encontrar um modelo global do ambiente percorrido por um robô móvel.

3 TRABALHOS RELACIONADOS

O constante desenvolvimento de pesquisas nas áreas de visão computacional e robótica móvel contribui para o aperfeiçoamento das soluções em várias operações, dentre elas o SLAM. Neste capítulo serão apresentados trabalhos relacionados ao tema desta dissertação de acordo com as respectivas categorias, mostrando o desenvolvimento e apontando tendências e caminhos para a evolução da área.

É importante ressaltar que a principal dificuldade e objetivo de pesquisas voltadas a visão computacional é conciliar o grande volume de informação que imagens disponibilizam com o tempo de processamento computacional necessário para a compreensão e conversão desta informação em dados úteis. De forma geral, os sistemas baseados em visão utilizam algum tipo de representação simplificada, a qual foca em um tipo específico de informação.

3.1 LOCALIZAÇÃO

DeSouza e Kak (2002) afirmam que uma vez que em localização absoluta a pose inicial do robô é desconhecida, o sistema de navegação deve construir uma correspondência entre as observações e as expectativas derivadas de toda a base de dados. Devido às incertezas associadas às observações, é possível que o mesmo conjunto de observações corresponda a múltiplas expectativas. As ambiguidades resultantes na localização podem ser resolvidas por métodos como: localização de Markov (THRUN, 2000), processos de Markov parcialmente observáveis (SIMMONS; KOENIG, 1995), localização de Monte Carlo (BLAKE; ISARD, 1997) (DELLAERT et al., 1999), filtros de Kalman com múltiplas hipóteses baseados em uma mistura de Gaussianas (COX; LEONARD, 1994), usando intervalos para representar incertezas (KROTKOV, 1989) (ATIYA; HAGER, 1993), e por triangulação determinística (SUGIHARA, 1988). Em um grande número de situações práticas, a posição inicial do robô é conhecida pelo menos aproximadamente. Nestes casos, o algoritmo de localização deve simplesmente acompanhar as incertezas na posição do robô quando executa comandos de movimento e, quando excederem um limiar, usar os sensores para uma nova correção em sua posição. Em geral, técnicas de localização probabilísticas evoluíram como a abordagem preferida para representação e atualização das incertezas de posicionais enquanto o

robô se move (DESOUZA; KAK, 2002).

Foletto et al. (2013) propõe um método estimador de estados não-linear intermitente para sistemas de controle via rede sem fio, onde o filtro de Kalman *Unscented* é utilizado para tratar o problema de estimação na tese. O método proposto foi aplicado em um sistema de teleoperação, um exemplo de aplicação real de um sistema de controle que utiliza uma rede de comunicação sem fio para interligar seus componentes.

Moore e Stouch (2014) desenvolveram um pacote de *software* chamado *robot_localization*, para o *Robot Operating System* (ROS). O qual contém uma implementação EKF de estimador de estados para fusão de sensores (*ekf_localization_node*). Esta implementação tem suporte para múltiplos sensores e alto nível de customização que fazem dela uma ótima solução para o problema de estimação de estados em uma variedade de plataformas robóticas. A Figura 14 mostra o resultado de uma das experiências realizadas por Moore e Stouch (2014) usando um robô móvel Pioneer 3 acoplado com sensores GPS e IMU. Posteriormente também foi desenvolvida uma implementação UKF no pacote pelos mesmos autores (*ukf_localization_node*).



Figura 14 – Saída do *ekf_localization_node* (verde) na fusão de dados de odometria, duas IMUs e duas unidades GPS. A trajetória do robô como uma média dos dados brutos dos GPS é mostrada em vermelho (MOORE; STOUCH, 2014).

3.2 SLAM E REGISTRO 3D

Endres et al. (2014) desenvolveram um moderno sistema de SLAM 3D para Sensores RGB-D, como o Microsoft Kinect. Sua abordagem extrai características (*keypoints*) visuais das imagens coloridas e usa as imagens de profundidade para localizá-las em 3D. RANSAC é utilizado para estimar as transformações entre *keypoints* associados e otimizar o gráfico de poses usando otimização não-linear. Finalmente, o mapa 3D volumétrico do ambiente é registrado em forma de nuvem de pontos ou *Octomap*, que podem ser usados para localização de robôs, navegação e planejamento de caminhos.

Labbé e Michaud (2014) criaram o RTAB-Map (*Real-Time Appearance-Based Mapping*) que é uma abordagem de RGB-D SLAM baseado em grafos com um detector incremental de fechamento de ciclo baseado em aparências. O detector de fechamento de ciclo (*loop closure detector*) usa uma abordagem *bag-of-words* para determinar a probabilidade de uma imagem nova provir de um local anterior ou de um novo local. Quando uma hipótese de fechamento de ciclo é aceita, uma nova restrição é adicionada ao gráfico do mapa e, em seguida, um otimizador de grafos minimiza os erros no mapa. Uma abordagem de gerenciamento de memória é usada para limitar o número de locais usados para detecção de fechamento de ciclo e otimização do grafo (LABBE; MICHAUD, 2013), para que as restrições em tempo real em ambientes de grande escala sejam sempre respeitadas. O RTAB-Map pode ser usado sozinho com um Kinect portátil ou câmera estéreo para mapeamento RGB-D em 6 DoF, ou em um robô equipado com um sensor de alcance a laser para mapeamento em 3 DoF, por exemplo. O RTAB-Map é embrulhado em um pacote do ROS chamado *rtabmap_ros*.

Em trabalhos relacionados com SLAM 3D, Huang et al. (2017) propõe um sistema de voo autônomo para odometria visual e mapeamento utilizando o processo EKF-SLAM com Kinect. Henry et al. (2010) propõe uma estrutura de mapeamento RGB-D que constrói mapas 3D densos de ambientes internos usando TORO, uma ferramenta de otimização desenvolvida para SLAM. Hervier, Bonnabel e Goulette (2012) construiu um mapa 3D preciso baseado na covariância estimada do ICP (*Iterative Closest Point*) e na fusão de dados do Kinect e um giroscópio por um EKF. Ling, Cheng e Burnett (2013) propõe a utilização do algoritmo EKF-SLAM de modo iterado, relinearizando a equação de medida por meio da iteração de um máximo aproximado a posteriori (MAP), que é estimado em torno no estado atualizado, ao invés de depender somente do estado previsto. Nas abordagens de Hu-

ang et al. (2017), Henry et al. (2010) e Hervier, Bonnabel e Goulette (2012), o EKF lineariza a predição de medida e todas as transformações no estado de predição usando expansão em séries, substituindo as matrizes Jacobianas por transformações lineares no filtro de Kalman. Entretanto, tais linearizações assumem que os erros de predição podem ser bem aproximados por uma função linear. Caso essa condição não possa ser satisfeita, os erros serão propagados e podem resultar diretamente na divergência das estimativas.

Sturm et al. (2012) apresentam um método moderno de avaliação de sistemas RGB-D SLAM. Foram gravados um grande conjunto de sequências de imagens de um Microsoft Kinect com poses da câmera precisas e sincronizadas (*ground truth*), obtidas através de um sistema de captura de movimento. As sequências contêm tanto imagens coloridas como de profundidade em resolução total do sensor (640 x 480) na frequência de quadros de vídeo (30Hz). A trajetória *ground truth* foi obtida através de um sistema de captura de movimento com oito câmeras de rastreamento de alta velocidade (100Hz). O conjunto de dados consiste em 39 sequências que foram registradas em um ambiente de escritório e um corredor industrial. O conjunto de dados abrange uma grande variedade de cenas e movimentos da câmera. As sequências são fornecidas para depuração com movimentos lentos bem como trajetórias mais longas com e sem fechamentos de ciclo. A maioria das sequências foram gravadas a partir de um Kinect portátil com movimentos não restritos de 6 DoF, mas também são fornecidas sequências de um Kinect montado em um robô Pioneer 3 que foi navegado manualmente através de um ambiente interno. Para estimular a comparação de diferentes abordagens, Sturm et al. (2012) fornecem ferramentas automáticas de avaliação tanto para a avaliação da deriva em sistemas de odometria visual quanto para o erro de pose global dos sistemas SLAM. Um site² criado contém todos os dados, descrições detalhadas das cenas, especificações dos formatos de dados, código de amostra e ferramentas de avaliação.

3.3 AVALIAÇÃO DE DETECTORES E DESCRIPTORES DE CARACTERÍSTICAS LOCAIS

Mikolajczyk e Schmid (2005) compararam o desempenho de detectores calculados para regiões de interesse. A avaliação utiliza como

²Disponível em: <<https://vision.in.tum.de/data/datasets/rgbd-dataset>>. Acesso em abr. 2018.

critério de comparação a precisão com relação ao *recall*, que é o número de correspondências corretas sobre número total de correspondências, e é realizada para diferentes transformações de imagem, comparando *shape context* (BELONGIE; MALIK; PUZICHA, 2002), filtros direcionáveis (*Steerable Filters*) (FREEMAN; ADELSON et al., 1991) (KE; SUKTHANKAR, 2004), PCA-SIFT, invariantes diferenciais (*differential invariants*) (KOENDERINK; DOORN, 1987), *spin images* (LAZEBNIK; SCHMID; PONCE, 2003), SIFT, filtros complexos (SCHAF-FALITZKY; ZISSERMAN, 2002), invariantes de momento (GOOL; MOONS; UNGUREANU, 1996), e correlação cruzada para diferentes tipos de regiões de interesse. Eles concluem que o desempenho dos descritores é em grande parte independente do detector e que os descritores baseados em SIFT possuem melhor desempenho. Miksik e Mikolajczyk (2012) compararam os *trade-offs* (relação de compromisso) entre precisão e velocidade de BRIEF, BRISK, ORB, LIOP (WANG; FAN; WU, 2011), MRRID e MROGH (FAN; WU; HU, 2012), fornecendo uma avaliação de detectores de características de desempenho e comparando sua respectiva precisão e velocidade em árvores k-d randomizadas e distância Hamming de busca por força bruta. Eles concluem que LIOP, MRRID, MROGH tem melhor desempenho que descritores SURF e SIFT em ambos os testes, precisão e *recall*, embora a sua eficiência seja baixa. Da mesma forma, descritores binários têm tempos de extração rápidos e baixas exigências de memória, proporcionando uma técnica eficiente para aplicações com limitações de tempo com boa precisão.

Bekele, Teutsch e Schuchert (2013) compararam os descritores binários BRIEF, ORB, BRISK e FREAK. Teste da razão (*ratio-test*) e RANSAC são utilizados para a exatidão, precisão e número médio de melhores correspondências como medidas de desempenho. Os autores concluem que a partir dos resultados obtidos, BRISK é altamente recomendável, possuindo a maior quantidade de melhores correspondências. Por outro lado, FREAK apresenta um desempenho levemente menor, mas com uma computação mais rápida.

Barbosa et al. (2017) apresentou uma comparação entre os descritores BRIEF, BRISK, FREAK, ORB, SIFT e SURF para verificar sua aplicabilidade no problema de navegação robótica através de quatro testes: Teste local; Teste global; Teste de velocidade de correspondência; Teste de dissimilaridade. Em seguida, Barbosa et al. (2017) avalia os algoritmos de localização e mapeamento propostos com os descritores BRIEF, BRISK, ORB, SIFT e SURF, através dos seguintes critérios de avaliação: Número de *keyframes*; *Relative Pose Error* (RPE); *Absolute*

Trajectory Error (ATE). Verificou-se que todos os descritores analisados, exceto o FREAK, atendem às características necessárias para o sistema de mapeamento e localização na forma que foi proposta. Os critérios de RPE e ATE indicam que os descritores BRISK, SURF e ORB obtêm erros relativamente menores do que os outros descritores.

3.4 CONCLUSÃO

De modo geral, o uso de descritores de características locais para comparar pontos de referência naturais (características locais) tem se mostrado como uma técnica satisfatória e de ampla utilização na área de visão computacional, encontrando resultados desejáveis e satisfatórios para as tarefas de localização e mapeamento.

O uso de técnicas de robótica probabilística em etapas de localização e mapeamento também é destaque em pesquisas recentes relacionadas à SLAM, em geral, apresentando resultados satisfatórios, com destaque para os algoritmos EKF e sua aplicação no problema de SLAM, EKF-SLAM.

A grande dificuldade enfrentada em SLAM 3D vem se mostrando ser o processamento e conciliação dos dados de imagem em tempo-real sem que haja uma perda significativa de detalhes.

4 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA REMOTO DE LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

Este capítulo apresenta a construção do sistema de localização e mapeamento remotos com registro 3D, com o objetivo principal de verificar a aplicabilidade de um sistema de localização e fusão de sensores probabilísticas, no caso um filtro de Kalman estendido, em um sistema de SLAM baseado em grafos para um robô móvel em um ambiente interno e plano tipo escritório. Os tópicos seguintes abordam as análises de requisito, bem como as respectivas características, recursos necessários, e estruturas de funcionamento do sistema propostos.

4.1 SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO

O algoritmo total proposto visa comandar um robô móvel de acionamento diferencial, no caso Pioneer 3-DX da ActivMedia, de forma remota e simultaneamente recolher dados provindos da odometria pelos *encoders*, leituras de varredura a laser (LiDAR 2D Sick LMS200) e imagens de cor e profundidade de uma câmera RGB-D, no caso um Microsoft Kinect, para realização remota do SLAM 3D com fusão de sensores. Para verificar a aplicabilidade do sistema foram feitos alguns testes que serão descritos ao longo deste capítulo.

4.1.1 Análise de requisitos

Os requisitos do sistema foram levantados a partir das necessidades encontradas para resolver o problema de localização na trajetória do robô ActivMedia Pioneer 3-DX simultaneamente a construção de uma representação em 3D do ambiente ao seu redor. Os requisitos funcionais e não-funcionais são descritos a seguir, assim como a forma de atuação do sistema.

4.1.1.1 Requisitos funcionais

RF-1. Escopo do sistema de localização e mapeamento remotos

A meta do sistema é realizar a localização do robô móvel durante sua trajetória de acordo com seu mapa através das imagens obtidas

por sua câmera. Os algoritmos de mapeamento e de localização devem ser capazes de se comunicar entre si, visto que o mapa proposto deve ser usado para inferir a localização, que por sua vez é utilizada na construção do mapa.

RF-2. Ambiente

Devido às restrições dos sensores e para simplificar o desenvolvimento, o ambiente utilizado pelo robô deverá ser plano, interno e bem iluminado. Restringindo, assim, o movimento do robô em 3 DoF.

RF-3. Odometria visual

O programa deve obter uma estimativa da trajetória do robô, isto é, um conjunto de poses absolutas, em tempo-real, através da configuração e utilização de um detector e descritor de características locais e do mapa sendo construído, de forma frequente e contínua.

RF-4. Localização por fusão de sensores

A parte específica referente à localização do robô deve estimar os estados de posição do robô pela fusão de 3 fontes de sensoriamento diferentes utilizando um método probabilístico e considerando uma distribuição Gaussiana de erro nas variáveis de estado do sistema. As fontes de sensoriamento são: Odometria pelos *encoders* nas rodas do robô; Odometria visual (Microsoft Kinect); e pose absoluta inferida pela varredura de um LiDAR 2D (Sick LMS200).

RF-5. Controle e visualização

Um operador deve enviar sinais de comando ao robô na forma de *SetPoints* de velocidade linear e angular, enquanto observa a construção do mapa remotamente.

4.1.1.2 Requisitos não-funcionais

RF-1. Composição e compatibilidade do mapa

O mapa deve conter características suficientes para localizar o robô sem possuir localizações ambíguas, atentando-se ao fato de que o robô pode retornar a uma área já visitada. Além disso, o mapa gerado deve ser capaz de ser aproveitado em outras tarefas do robô como planejamento de trajetórias e localização de objetos.

RF-2. Tempo de *setup*

Deve ser respeitado o tempo de inicialização do robô, dos sensores e aplicações referentes ao sistema, bem como as condições de operação dos sensores de modo que funcionem da maneira esperada.

RF-3. Consumo de banda

A frequência de fluxo de imagens do Kinect deve ser limitada,

de modo que o consumo de banda da rede seja reduzido à cerca de 500 KiB/s de largura de banda. Assim, assegurando o fluxo contínuo de imagens vindas do robô sem travamentos.

RF-4. Tempo de correspondência

O sistema deve ser capaz de realizar a construção do mapa 3D de modo que o operador possa visualizar simultaneamente a representação do mapa sob a perspectiva do robô enquanto controla-o remotamente.

RF-5. Confiabilidade da localização

A confiabilidade do sistema de localização deve existir para qualquer imagem da trajetória do robô. O sistema deve ser apto a resolver o problema de localização global, incluindo o problema do robô seqüestrado e do robô despertado.

4.1.2 Recursos

4.1.2.1 Software

O ROS é uma estrutura flexível para escrever software robótico. É uma coleção de ferramentas, bibliotecas e convenções que visa simplificar a tarefa de criar um comportamento robótico complexo e robusto em uma ampla variedade de plataformas. ROS está sob uma licença *open-source* e já existem diversos pacotes desenvolvidos pela comunidade para diversas finalidades, e.g. localização, SLAM, Kinect e controle do robô, que foram utilizados para o desenvolvimento deste projeto, entre eles: *RosAria*, *freenect_launch*, *sicktoolbox_wrapper*, *laser_scan_matcher*, *robot_localization*, *rtabmap_ros*, *teleop_twist_keyboard* e suas respectivas dependências. ROS tem suporte para as linguagens de programação Python e C++. Com exceção do pacote *teleop_twist_keyboard*, programado em Python, todos os outros pacotes citados foram desenvolvidos em C++.

Somente os pacotes *RosAria*, *freenect_launch*, *sicktoolbox_wrapper*, com suas bibliotecas e dependências foram embarcados no robô. Os demais softwares foram executados em um computador remoto.

A formulação do EKF é bem conhecida na literatura, o modelo utilizado por Moore e Stouch (2014) no desenvolvimento do pacote *robot_localization*, usado neste projeto, será detalhado aqui para mostrar os detalhes importantes da implementação. O objetivo é estimar a pose 3D completa (6 DoF) e a velocidade de um robô móvel ao longo do tempo. Este processo pode ser descrito como um sistema dinâmico não-linear, descrito por

$$\mathbf{s}_k = f(\mathbf{s}_{k-1}) + \mathbf{w}_{k-1}, \quad (4.1)$$

onde s_k é o vetor de estados do sistema (i.e., a posição e orientação 3D do robô) no tempo k , f é uma função não-linear de transição de estados e w_{k-1} é o ruído no processo, o qual é considerado ser distribuído normalmente (distribuição Gaussiana). No pacote *robot_localization* é possível configurar um vetor de estados de até 15 dimensões, composto pela posição 3D do veículo, orientação 3D, suas respectivas velocidades lineares e angulares e acelerações lineares. Valores angulares são expressados em ângulos de Euler (rolagem (α), arfagem (β) e guinada (γ)). Adicionalmente são recebidas medidas na forma

$$\mathbf{z}_k = h(\mathbf{s}_k) + \mathbf{v}_k, \quad (4.2)$$

onde z_k é a medida no tempo k , h é o modelo não-linear do sensor que mapeia os estados no espaço de medição e v_k é o ruído de medição normalmente distribuído.

O primeiro passo do algoritmo é o da predição, mostrado nas equações (4.3) e (4.4), que projeta o estado atual estimado e suas covariâncias adiante no tempo:

$$\hat{\mathbf{s}}_k = f(\mathbf{s}_{k-1}). \quad (4.3)$$

$$\hat{\mathbf{P}}_k = \mathbf{F} \mathbf{P}_{k-1} \mathbf{F}^T + \mathbf{Q}. \quad (4.4)$$

Nessa aplicação, f é um modelo cinemático padrão derivado da mecânica Newtoniana. A covariância do erro estimada, $\hat{\mathbf{P}}$, é projetada por \mathbf{F} , a matriz Jacobiana de f e, em seguida, perturbada por \mathbf{Q} , a covariância do ruído do processo.

O próximo passo, de correção, é mostrado nas equações (4.5) a (4.7):

$$\mathbf{K} = \hat{\mathbf{P}}_k \mathbf{H}^T (\mathbf{H} \hat{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R})^{-1}. \quad (4.5)$$

$$\mathbf{s}_k = \hat{\mathbf{s}}_k + \mathbf{K} (\mathbf{z} - \mathbf{H} \hat{\mathbf{s}}_k). \quad (4.6)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K} \mathbf{H}) \hat{\mathbf{P}}_k (\mathbf{I} - \mathbf{K} \mathbf{H})^T + \mathbf{K} \mathbf{R} \mathbf{K}^T. \quad (4.7)$$

Primeiramente, o ganho de Kalman \mathbf{K} é calculado a partir da matriz de observação \mathbf{H} , a covariância das medidas \mathbf{R} e $\hat{\mathbf{P}}_k$. Em seguida, o ganho é utilizado para atualizar o vetor de estados e a matriz de covariâncias. A equação de atualização de covariâncias (4.7) é empregada para garantir a estabilidade do filtro, assegurando que \mathbf{P}_k permaneça positiva semi-definida (MOORE; STOUCH, 2014).

Pela formulação padrão do EKF, \mathbf{H} deve ser uma matriz Jacobiana da função do modelo de observação h . Para suportar uma ampla gama de sensores, assume-se que cada sensor produz medições das variáveis de estado que estão sendo estimadas. Assim, \mathbf{H} é simplesmente uma matriz identidade. O filtro suporta atualizações parciais do vetor de estados, o que é necessário pois geralmente os sensores não disponibilizam todas as variáveis do vetor de estados. Na prática, isso é feito através de \mathbf{H} . Por exemplo, ao se medir apenas m variáveis, \mathbf{H} torna-se uma matriz m por 15 de rank m , com valores de 1 e 0 nas colunas de variáveis medidas e não medidas, respectivamente.

4.1.2.2 Hardware

Com relação aos recursos de hardware, como dito anteriormente, foram utilizados o robô Pioneer 3-DX, uma câmera RGB-D Microsoft Kinect, um LiDAR 2D Sick LMS200 e um computador remoto com processador Intel CORE i5. A plataforma robótica montada pode ser observada na Figura 15.

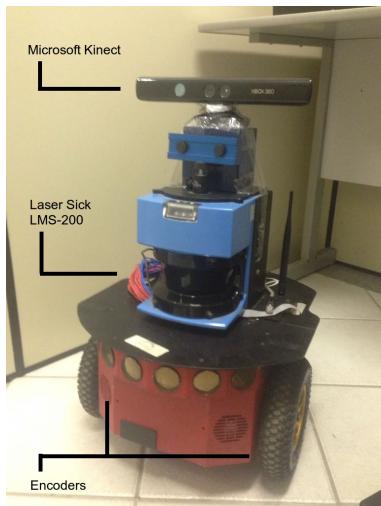


Figura 15 – Foto do robô Pioneer 3-DX montado com os sensores usados no projeto.

Os sistemas de acionamento Pioneer 3 usam motores DC reversíveis de alta rotação e alto torque, cada um equipado com um

codificador de quadratura ótica de alta resolução no eixo para detecção precisa de posição e velocidade e cálculo avançado de ponto morto (*dead-reckoning*). Pulsos por revolução e tamanhos de pneus variam de acordo com o modelo do robô. No entanto, o programa pode corrigir erros de correspondência de pneus e converter a maioria dos comandos do cliente e informações do servidor remotamente através dos parâmetros *DriftFactor*, *TicksMM* e *RevCount*. Os valores padrões destes parâmetros, usados neste projeto, são mostrados na tabela 1. O servidor de acionamento dos motores utiliza um comum controlador proporcional-integral-derivativo (PID) com realimentação roda-*encoder* para ajustar um sinal de modulação de largura de pulso (PWM) nos acionadores do motor para controlar a potência dos motores. Os pneus são sólidos e cheios de espuma.

Tabela 1 – Valores padrões dos parâmetros do P3-DX.³

<i>TicksMM</i>	<i>DriftFactor</i>	<i>RevCount</i>	<i>SIPCYCLE</i>
128	0	16570	100

TicksMM é o número de pulsos do *encoder* por milímetro de rotação do pneu para cálculos de velocidade de translação, rotação e distância. *DriftFactor* é um valor sinalizado em incrementos de 1/8192 que é adicionado ou subtraído nos pulsos do *encoder* da roda esquerda para corrigir as diferenças de circunferência dos pneus e consequentemente o desvio de translação e rotação. O parâmetro *RevCount* é o número diferencial de pulsos dos *encoders* para uma rotação de 180 graus do robô. O *SIPCYCLE* informa o número de milissegundos entre dois pacotes de informação do servidor consecutivos do robô.

Assim, um erro na determinação da velocidade pode ser resultado da falta de um pulso do *encoder* em um ciclo. Isso resultaria em um erro de velocidade igual a 1/*TicksMM*/*SIPCYCLE* (mm/ms ou m/s) para cada roda. Para os parâmetros do P3-DX mostrados na Tabela 1, este valor é $7,8125e^{-5}$. Note que ocorreria um erro de mesma quantidade absoluta de velocidade no próximo ciclo. A caixa de engrenagens também desempenha um papel no erro de velocidade, mas seria necessária uma medição para encontrar a quantidade exata. Os valores de covariância na odometria pelos *encoders* não são fornecidos pelo pacote *RosAria*.

O LiDAR 2D LMS200 é uma verão de curto alcance para aplicações em ambientes internos. Alguns dados de performance e caracte-

³Dados fornecidos pelo fabricante.

rísticas do LiDAR 2D LMS200 são mostrados na Tabela 2.

Tabela 2 – Performance e características do LMS200. ⁴	
Fonte de luz	Infravermelho (905 nm)
Ângulo de abertura	180°
Frequência de varredura	75 Hz
Faixa de trabalho	0 m ... 80 m
Alcance com 10% de remissão	10 m
Tempo de resposta	≥ 13 ms
Erro sistemático típico	±15 mm
Erro estatístico típico	±5 mm

O sistema de detecção de profundidade do Microsoft Kinect consiste no emissor de laser IR e na câmera de infravermelho. Um padrão conhecido de pontos é projetado a partir do emissor de laser IR. Estes pontos são gravados pela câmara de IR e depois comparados com o padrão conhecido. Quaisquer perturbações são conhecidas por serem variações na superfície e podem ser detectadas como mais próximas ou mais afastadas. Essa técnica é conhecida como luz estruturada. A câmara IR opera a 30 Hz e possui resolução de 1200x960 pixels. Estas imagens são sub-amostradas a 640x480 pixels com 11-bits, o que fornece 2048 níveis de sensibilidade. A câmara RGB, que opera a 30 Hz, possui resolução em 640x480 pixels com 8 bits por canal. O Kinect também tem a opção de mudar a câmara para alta resolução, executando em 10 FPS em 1280x1024 pixels. A própria câmara possui um conjunto de recursos, incluindo balanceamento automático de branco, referência preta, evitação de cintilação, saturação de cor e correção de defeitos (CRUZ; LUCIO; VELHO, 2012).

A Tabela 3 apresenta algumas especificações em relação ao Microsoft Kinect v1, usado neste projeto, e a Figura 16 mostra seus componentes internos. O dispositivo também conta com um motor de inclinação e uma unidade de medida inercial (IMU) de 3 eixos, não utilizados neste projeto.

Outro desafio nos dados de imagem que os usuários devem ter em mente é que o Kinect usa um obturador rolante para a câmara que pode levar a distorções de imagem quando a câmara é movida rapidamente (STURM et al., 2012). No caso deste projeto, foi improvisado um amortecedor para amenizar tais distorções nas imagens. Além disso, o movimento do Kinect deve ser compatível com a taxa de captura,

⁴Dados fornecidos pelo fabricante.

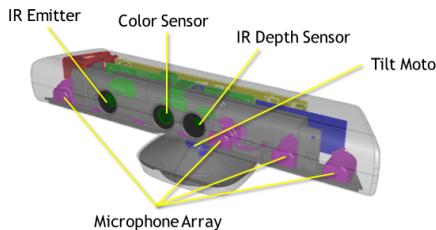


Figura 16 – Imagem dos componentes internos do Kinect. Fonte: Microsoft Developer Network.

Tabela 3 – Especificações do Microsoft Kinect v1.⁵

Câmera colorida	640x480 pixels com 30 FPS
Câmera IR	Sensor de 1280x1024 pixels, saída de 640x480 pixels com 30 FPS
Faixa de operação	0,8 m ... 3,5 m
Campo de visão	58° H, 45° V, 70° D
Resolução espacial (à 2 m de distância)	3 mm
Resolução de profundidade (à 2 m de distância)	1 cm

grandes e rápidas mudanças de direções podem causar nuvens de pontos sequenciais distantes entre si, o que dificulta o processo de registro do mapa.

4.1.3 Avaliação Experimental

Primeiramente foram analisados os métodos para obtenção da odometria visual a partir das imagens fornecidas pelo Kinect. Dentro as diversas possibilidades de configuração, a odometria visual e o dicionário visual foram configurados com o detector GFTT (*Good Features To Track*) para detectar as características na imagem e BRIEF (*Binary Robust Independent Elementary Features*) de 16 bytes como descritor binário dessas características. Este descritor binário pode ser usado sob a licença BSD. Embora menos descriptivo do que os descritores SIFT e SURF (tipo *float*), para ambientes pequenos funciona bem.

⁵Dados fornecidos pelo fabricante.

Em ambientes de grande escala, SURF ou SIFT são mais recomendados. Em testes realizados por Labbé e Michaud (2017), o modelo GFIT+BRIEF ofereceu maior confiabilidade entre todos os detectores de características binários disponíveis. Segundo eles, os detectores FAST ou ORB são mais rápidos que a GFIT, mas geram menos características boas para rastrear e os descritores FREAK e BRIEF fornecem resultados semelhantes.

Em seguida foram gravados sequências de dados das três fontes de sensoriamento em relação à uma mesma movimentação do robô e essas sequências de dados foram analisadas para a melhor configuração do vetor de estados do sistema e a seleção de quais variáveis de estado de cada sensor serão fundidas no processo de localização EKF. Devido às restrições por parte da odometria via *encoders* (não são medidas as variáveis z , rolagem (α), arfagem (β) e suas respectivas velocidades e aceleração) e da pose do laser (não são medidas as variáveis z , rolagem (α), arfagem (β)), o ambiente selecionado para a tarefa foi um ambiente plano, interno e com boa luminosidade, no qual foram desconsideradas as eventuais variações sutis no solo. Em razão disto, mapeamento, odometria visual e EKF foram restringidos ao modo 2D, zerando os valores de z , rolagem (α), arfagem (β) e suas respectivas velocidades e aceleração. A posição inicial do robô foi considerada a origem do sistema de coordenadas.

Tabela 4 – Tabela com as variáveis do espaço de estados que são medidas pelos sensores e as utilizadas para fusão pelo EKF.

Sensores	Variáveis disponíveis	Variáveis utilizadas
Odometria via encoders	$x, y, \gamma, \dot{x}, \dot{\gamma}$	$\dot{x}_{dif}, \dot{\gamma}_{dif}$
Odometria visual	$x, y, \gamma, \dot{x}, \dot{\gamma}$	$\dot{x}, \dot{\gamma}$
Pose via laser	x, y, γ	x, y, γ

É importante ressaltar que na teoria espera-se que os sensores reportem devidamente suas covariâncias em cada medição, mas na prática isso pode não ocorrer, como no caso deste projeto, em que os valores de odometria obtidos pelos *encoders* são reportados com covariâncias nulas, indicando uma medida sem erros que é impossível na prática. Para contornar esses casos, o algoritmo assume valores de covariância baixos (10^{-6}) e diferencia os valores absolutos de posição e orientação (x, y, γ) para obter valores relativos de velocidade linear e angular ($\dot{x}, \dot{\gamma}$) que, então, são fundidos no filtro, deste modo, as covariâncias ganham

uma pequena quantidade de erro a cada iteração. Sendo assim, o vetor de estados definido e configurado para filtro foi:

$$\mathbf{x}_k = [x, y, \gamma, \dot{x}, \dot{\gamma}]^T \quad (4.8)$$

Por fim, a mesma sequencia de dados foi repetida diversas vezes implementando o sistema de localização EKF para refinar, de modo empírico, a configuração da matriz de covariância dos ruídos de processo \mathbf{Q} , usada para modelar a incerteza no estágio de previsão dos algoritmos de filtragem, e da matriz de covariância estimada inicial \mathbf{P}_0 , que define o erro estimado no estado inicial. Segundo Moore e Stouch (2014), como a covariância do ruído do processo pode ser difícil de ajustar para uma determinada aplicação, o *ekf_localization_node* expõe essa matriz como um parâmetro para os usuários, permitindo um nível adicional de personalização. As configurações finais obtidas foram:

$$Q = \begin{pmatrix} 0,05 & 0 & 0 & 0 & 0 \\ 0 & 0,06 & 0 & 0 & 0 \\ 0 & 0 & 0,06 & 0 & 0 \\ 0 & 0 & 0 & 0,03 & 0 \\ 0 & 0 & 0 & 0 & 0,03 \end{pmatrix}; \text{ e}$$

$$P_0 = \begin{pmatrix} 1e^{-6} & 0 & 0 & 0 & 0 \\ 0 & 2e^{-6} & 0 & 0 & 0 \\ 0 & 0 & 1e^{-6} & 0 & 0 \\ 0 & 0 & 0 & 2e^{-7} & 0 \\ 0 & 0 & 0 & 0 & 2e^{-7} \end{pmatrix}.$$

4.1.4 Estrutura do algoritmo

A Figura 17 apresenta um diagrama simplificado da estrutura do sistema proposto neste trabalho utilizando a plataforma mostrada na Figura 15 e um computador remoto ligados em uma rede sem fio padrão, ambos com sistema operacional Ubuntu 14.04 LTS (*Trusty Tahr*) e ROS Indigo.

Comandos de velocidade linear e angular ($\dot{x}_{sp}, \dot{\gamma}_{sp}$) são enviados ao robô via teclado, a velocidade do robô foi limitada para facilitar o processamento da odometria visual, os valores de velocidade linear e angular definidos são:

$$\dot{x}_{sp} = \pm 0,8 \text{ [m/s]; e}$$

$$\dot{\gamma}_{sp} = \pm 0,2 \text{ [rad/s].}$$

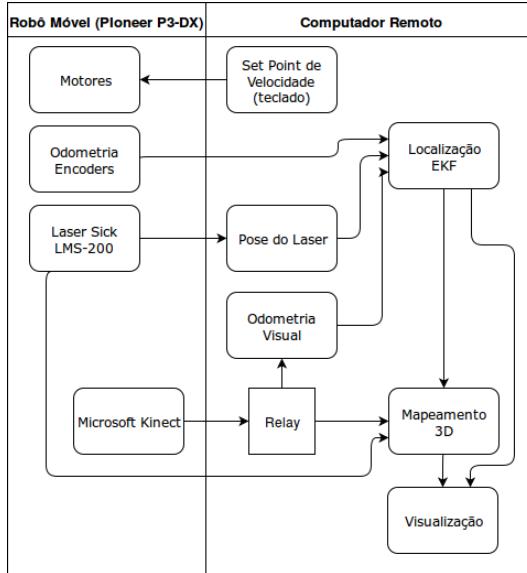


Figura 17 – Diagrama simplificado do sistema de localização e mapeamento remotos usado no projeto.

O pacote *RosAria* se encarrega de enviar esses valores ao controlador interno dos motores do robô e converter as leituras dos *encoders* em valores de odometria. Os escaneamentos feitos pelo laser são enviados ao computador (do pacote *sicktoolbox_wrapper* ao *laser_scan_matcher*) que compara as varreduras para obtenção de uma pose absoluta estimada do laser, usamos os valores de odometria dos *encoders* para fornecer uma suposição para a posição atual do sensor toda vez que uma nova mensagem de varredura chegar, aumentando a velocidade de conversão e precisão do processo de estimação. Quando não há nenhum palpite disponível, uma suposição razoável (e amplamente usada) é que o sensor não se moveu (modelo de velocidade zero). O pacote *freenect_launch*, que possui as bibliotecas e *drivers* para comunicação com o Kinect, envia com uma frequência reduzida de *5Hz* as imagens RGB e D ao computador que republica essas imagens através de um *Relay*, que foi utilizado para aliviar o consumo de banda da rede pois vários nós usam as mesmas imagens para suas funcionalidades (Odometria visual, mapeamento e visualização).

A fusão de sensores pelo algoritmo EKF foi aplicada pelo pacote *robot_localization*, que foi detalhada anteriormente na subseção de re-

cursos. O filtro foi programado com uma frequência de 30Hz, sendo que a odometria via *encoders* têm um frequência de 20Hz, odometria visual opera em cerca de 5Hz e a pose via laser 10Hz, aproximadamente.

As etapas de odometria visual, mapeamento 3D e visualização são feitas pelo pacote *rtabmap_ros*, desenvolvido por Labb   e Michaud (2017),    uma abordagem RGB-D SLAM baseada em grafos com um detector incremental de fechamento de ciclo baseado em apar  ncias.

Um sistema rob  tico normalmente possui muitos quadros de coordenadas 3D que mudam com o tempo, como um quadro global, quadro base, quadro da preens  o, quadro da cabe  a, etc. O ROS utiliza *tf* para monitorar todos esses quadros ao longo do tempo e permite fazer perguntas como:

- Onde estava o quadro da cabe  a em rela  o ao quadro global, 5 segundos atr  s?
- Qual    a pose do objeto no manipulador em rela  o    base do rob  o?
- Qual    a pose atual do quadro base no quadro do mapa?

tf pode operar em um sistema distribu  do. Isso significa que todas as informa  es sobre os quadros de coordenadas de um rob  o est  o dispon  veis para todos os componentes do ROS em qualquer computador no sistema. N  o h   servidor central de transforma  o. A arvore *tf* de frames que foi obtida neste projeto encontra-se em ap  ndice.

Um diagrama completo do sistema, com todos os pacotes, n  os, t  picos e servi  os configurados e utilizados pelo ROS, bem como suas rela  es, pode ser obtido atrav  s do pacote *rqt_graph* para uma an  lise mais complexa do sistema.

4.1.5 Crit  rios de Avalia  o

Um m  todo ideal de avalia  o dos algoritmos pode ser comparando-se as poses obtidas na etapa de localiza  o com o valor da pose real, obtida atrav  s de algum sistema de localiza  o altamente preciso, denominado *ground truth*. Sturm et al. (2012) propõe duas m  tricas de avalia  o: o erro relativo de pose (*Relative pose error* (RPE)) e o erro absoluto de trajet  ria (*Absolute trajectory error* (ATE)).

O erro de pose relativo na etapa de tempo i    definido como:

$$E_i := (Q_i^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}) \quad (4.9)$$

O erro de pose relativo mede a precisão local da trajetória ao longo de um intervalo de tempo fixo Δ . Com base em uma sequência de n poses da câmera, obtém-se dessa maneira $m = n - \Delta$ erros de pose relativos individuais ao longo da sequência. A partir desses erros, Sturm et al. (2012) propõe calcular o erro quadrático médio (RMSE) sobre todos os índices de tempo da componente translacional como:

$$RMSE(E_{i:n}, \Delta) := \sqrt{\left(\frac{1}{m} \sum_{i=1}^m ||trans E_i||^2 \right)} \quad (4.10)$$

Onde $trans E_i$ refere-se aos componentes de translação do erro de pose relativo E_i . Deve-se notar que alguns pesquisadores preferem avaliar o erro médio em vez do erro quadrático médio que dá menos influência aos valores atípicos. O erro de rotação também pode ser avaliado, embora Sturm et al. (2012) afirme que a comparação por erros de translação é suficiente, uma vez que os erros de rotação estão implícitos e podem ser observados indiretamente (visto que erros de rotação aparecem como erros de translação quando a câmera é movida).

De forma complementar, é interessante analisar a consistência global da trajetória estimada, podendo ser avaliada pela comparação entre as distâncias absolutas da trajetória estimada e da trajetória objetivo. Como ambas as trajetórias podem ser especificadas em quadros de coordenadas arbitrárias, elas precisam primeiro ser alinhadas. Isto pode ser concluído usando o método de Horn (HORN, 1987), que encontra a transformação de corpo rígido S correspondente à solução de mínimos quadrados que mapeia a trajetória estimada $P_{1:n}$ na trajetória real no solo $Q_{i:n}$. Dada esta transformação, o erro absoluto da trajetória no tempo i pode ser calculado como:

$$F_i := Q_i^{-1} S P_i \quad (4.11)$$

Similarmente ao erro relativo de pose, Sturm et al. (2012) propõe avaliar o erro médio quadrático sobre todos os índices de tempo dos componentes de translação, isto é:

$$RMSE(F_{i:n}, \Delta) := \sqrt{\left(\frac{1}{m} \sum_{i=1}^m ||trans F_i||^2 \right)} \quad (4.12)$$

Para essas avaliações, assume-se que tem-se uma sequência de

n poses a partir da trajetória estimada e outra sequência a partir da trajetória verdadeira. Para simplificar a notação, será assumido que as sequências são sincronizadas no tempo, igualmente amostradas, e ambas têm comprimento n .

Na prática, independentemente do sensor usado, estas duas sequências têm taxas de amostragem e comprimentos diferentes, possuindo dados faltantes, de modo que é necessário um passo adicional de associação e interpolação de dados.

Pela falta de um método sensorial de alta performance para obtenção do *ground truth* (como o conjunto de câmeras de alta resolução utilizadas por Sturm et al. (2012), por exemplo), assim como pela falta dos dados de leitura de laser e odometria nos arquivos de dados disponibilizados por Sturm et al. (2012) (o que torna os dados inaplicáveis no sistema aqui proposto), as poses obtidas através do sensor LiDAR foram consideradas como *ground truth*, já que dentre os sensores disponíveis este é o mais preciso.

Além disso, serão plotados gráficos das variáveis de estado do sistema e suas covariâncias para análise visual, bem como mostrado um exemplo do registro final 3D em nuvem de pontos e grades de ocupação 2D inferidas.

5 EXPERIMENTOS E RESULTADOS

No experimento inicial, o robô foi comandado para percorrer uma trajetória pequena, os dados dos sensores e a saída do filtro foram gravados em um arquivo *.bag* e então foram plotados gráficos via *Matlab* das variáveis de estado e covariâncias relevantes para análise. Os resultados podem ser observados nas Figuras 18 a 21.

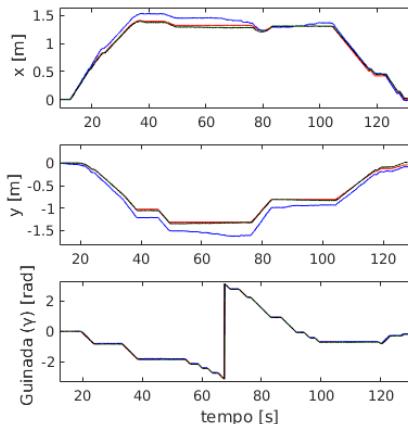


Figura 18 – Gráfico da posição (x, y) e orientação (γ) absolutas reportadas pelos sensores: Odometria encoders (vermelho), Odometria visual (azul), pose do laser (verde). E o resultado da fusão pelo EKF (preto).

Como pode ser observado na Figura 18, os sensores reportam valores semelhantes e o EKF realiza bem a fusão dos sensores no sistema, sendo os valores estimados bem próximos dos valores reportados pelo laser. Apesar do filtro não usar os valores absolutos reportados pelos *encoders* e odometria visual, esses valores foram plotados para efeito de comparação, assim como na Figura 20.

Na Figura 19, são mostrados os valores de velocidades usados pelo filtro. Apesar das oscilações, o filtro consegue convergir. Esses valores não são usados no mapeamento, onde somente a posição absoluta do robô é utilizada.

A Figura 20 mostra a trajetória do robô vista de cima. Como pode ser observado, a trajetória obtida pelo EKF é bem semelhante a trajetória reportada pelo laser e os valores obtidos pela odometria

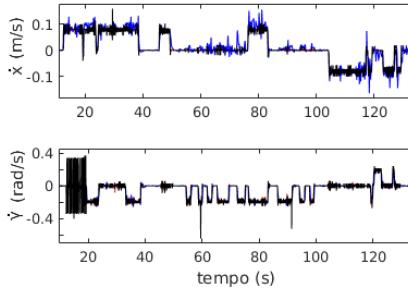


Figura 19 – Gráfico das velocidade linear (\dot{x}) e angular (\dot{y}) relativas reportados pelos sensores: Odometria encoders (vermelho), Odometria visual (azul). E o resultado da fusão pelo EKF (preto).

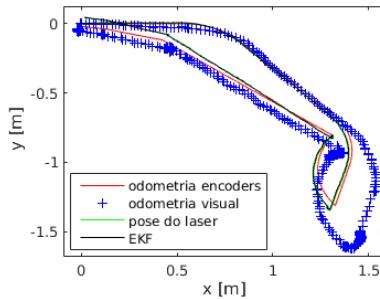


Figura 20 – Gráfico da trajetória (x, y) do robô vista de cima. As legendas se encontram na figura.

visual se mostram maiores em relação aos outros sensores, fato que pode ser reparado através de uma calibração mais refinada do Kinect, porém, neste caso, é interessante a presença destes erros para testar a estimativa do filtro.

As covariâncias das variáveis relevantes ao sistema são mostradas na Figura 21. Para melhor visualização, as covariâncias em x, y e γ foram ampliadas em torno de $t = 80s$. Caso o filtro não tivesse nenhuma fonte de medição absoluta (x, y ou γ), as covariâncias dessas variáveis iriam crescer sem limites, podendo levar à instabilidade. Neste projeto foram utilizadas as medições de posição absoluta obtidas pelo laser (como mostrado na Tabela 1), pelo fato deste ser o sensor com maior precisão entre os três.

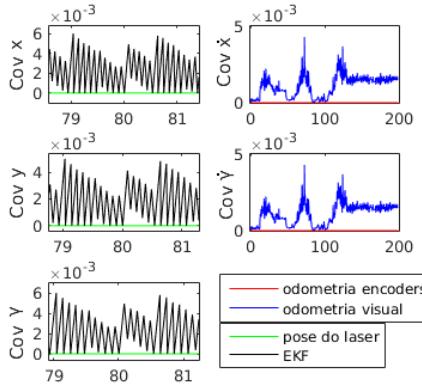


Figura 21 – Gráfico das covariâncias reportadas pelos sensores e da fusão pelo EKF pelo tempo. As legendas se encontram na figura.

A Figura 22 apresenta o comportamento do filtro quando ocorre uma falha na odometria visual, ou seja, quando não estão disponíveis características suficientes na imagem para um cálculo de transformação do movimento e, assim, impossibilitando o cálculo da posição ou orientação da câmera, isso pode ocorrer devido à movimentações angulares muito rápidas do robô, por exemplo. Neste caso, a odometria visual reporta valores nulos de posição e orientação no intervalo de tempo de 14 à 23 segundos, assim como suas velocidades, porém com valores bem altos de covariância (10000), o que faz com que o filtro ignore essas medições.

Com base nos dados obtidos, foi utilizada a ferramenta de avaliação automática criada por Sturm et al. (2012) para calcular o erro relativo de pose (*Relative pose error* (RPE)) e o erro absoluto de trajetória (*Absolute trajectory error* (ATE)). Como visto anteriormente, o ideal seria um sistema de localização altamente preciso para realizar a comparação com o sistema aqui proposto, porém, pela falta de tal recurso e pela necessidade de uma análise quantitativa do sistema, os valores de pose obtidos pelo LiDAR foram considerados como *ground-truth* por este ser o sensor mais preciso disponível. Os algoritmos em *python evaluate_rpe.py*⁶ *evaluate_ate.py*⁶ e , criados e disponibilizados por Sturm et al. (2012), foram aplicados e os resultados obtidos estão nas Tabelas 5 e 6 e nas Figuras 23 e 24.

⁶Disponível

em:

<<https://svncvpr.in.tum.de/cvpr-ros>

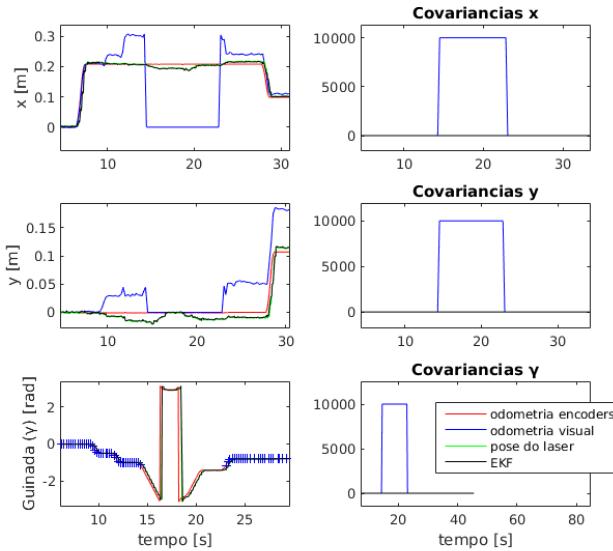


Figura 22 – Gráfico com falha na odometria visual. As legendas se encontram na figura.

Tabela 5 – *Relative pose error* (RPE) [m]

TRMSE	TMAX
0,004484	0,025275

A Tabela 5 apresenta os resultados do critério RPE em metros, onde os erros RMSE e máximo são apresentados como RMSE e MAX. A letra T antes de RMSE e MAX indica referência aos erros translacionais. O RPE mede a diferença entre o movimento estimado e o movimento real e pode ser usado para avaliar o erro global de uma trajetória por média em todos os intervalos de tempo possíveis. Os erros translacionais baixos obtidos, indicam um bom desempenho do sistema. A Figura 23 apresenta um gráfico do erro translacional obtido através do algoritmo.

Em vez de avaliar diferenças de poses relativas, o ATE primeiro alinha as duas trajetórias e, em seguida, avalia diretamente as dife-

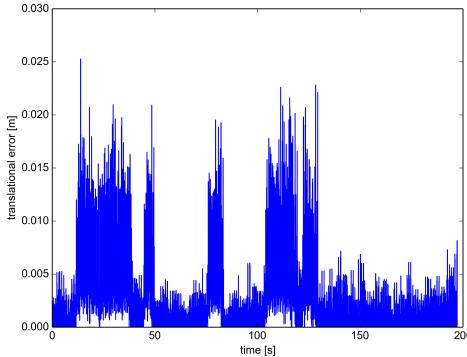


Figura 23 – Gráfico do erro translacional obtido pelo algoritmo *evaluate_rpe.py*.

Tabela 6 – *Absolute trajectory error* (ATE) [m]

RMSE	MIN	MAX
0,000412	0	0,008606

renças de pose absolutas. Os erros obtidos RMSE, mínimo e máximo, mostrados na Tabela 6, retificam as conclusões observadas através do RPE, mostrando um bom desempenho do sistema. A Figura 24 mostra a comparação de trajetórias realizada pelo algoritmo.

Em seguida, dois mapas completos do ambiente selecionado foram feitos, um deles utilizando o EKF para fusão de sensores e outro usando somente odometria visual. As grades de ocupação geradas através do pacote *rtabmap_ros* são exibidas na Figura 25, foi utilizado este tipo de representação 2D pois, desta forma, fica mais fácil a comparação de tamanho dos mapas. Observa-se que ambos os métodos proporcionaram um mapa bem representativo do ambiente, com um bom fechamento de ciclo (*loop closure*). É possível notar um pequeno deslocamento angular em uma das salas do mapa obtido usando EKF devido à uma colisão acidental do robô com uma parede, indicando uma pequena desvantagem do sistema (*encoders* reportaram movimento quando o robô estava parado). Contudo, foi possível localizar o robô normalmente quando ele saiu desta sala. O mapa obtido exclusivamente pela odometria visual também apresenta pequenas imperfeições e nota-se que ele é um pouco maior.

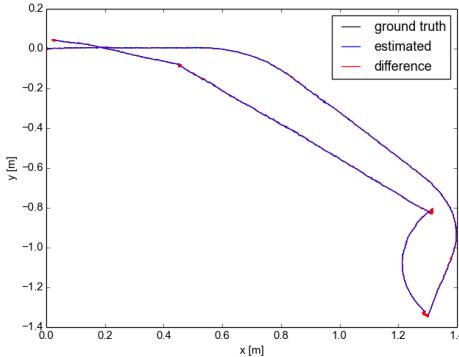


Figura 24 – Comparaçāo de trajetórias obtida pelo algoritmo *evaluate_ate.py*. As legendas se encontram na imagem.

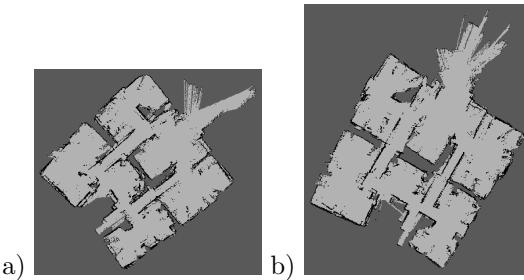


Figura 25 – Grades de ocupação geradas pelo pacote *rtabmap_ros* nos casos: a) Localizaçāo EKF; b) Odometria visual.

Por fim, a Figura 26 apresenta uma visão em perspectiva do mapa 3D em nuvem de pontos gerado pelo pacote *rtabmap_ros* com a localizaçāo feita pelo EKF no primeiro experimento. E a Figura 27 mostra um foto tirada de uma perspectiva semelhante por uma câmera convencional para comparação visual.

5.1 CONFIGURAÇĀO ALTERNATIVA

Durante a realização dos experimentos notou-se que, apesar de ser o sensor mais preciso, o sistema de localizaçāo utilizando o LiDAR necessita de características suficientemente disponíveis para comparar

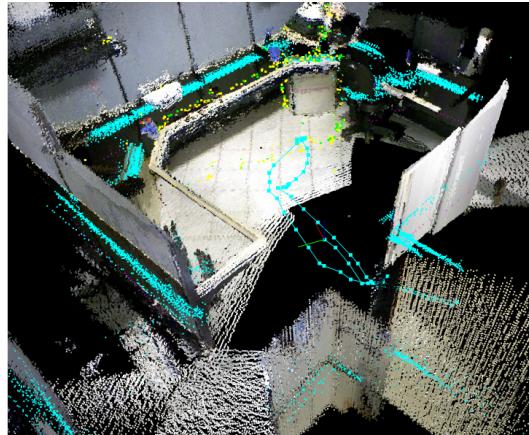


Figura 26 – Visualização em perspectiva do mapa 3D de nuvem de pontos gerado pelo pacote *rtabmap_ros* (trajetória do robô em azul).

um escaneamento com outro. Em alguns casos, na movimentação linear paralela ás paredes de um corredor, a pose do robô inferida pelo LIDAR, pela falta de características, se manteve convergindo para um ponto de máximo local, evidenciando a maior desvantagem do sistema proposto, que é a fusão de valores de medida equivocados e covariâncias reportadas incorretamente que influenciam na estimativa final de estados. O caso descrito pode ser observado pela Figura 28 .

Com base nessas observações, uma nova configuração do EKF foi aplicada, com as variáveis de estado dos sensores selecionadas de acordo com a Tabela 7, onde os valores absolutos da odometria via *encoders* e pose via laser foram diferenciados para obtenção dos valores relativos \dot{x}, \dot{y} .

Tabela 7 – Tabela com a configuração alternativa das variáveis do espaço de estados que são utilizadas para fusão pelo EKF.

Sensores	Variáveis utilizadas
Odometria via encoders	$\dot{x}_{dif}, \dot{y}_{dif}$
Odometria visual	\dot{x}, \dot{y}
Pose via laser	$\dot{x}_{dif}, \dot{y}_{dif}$



Figura 27 – Foto em perspectiva do ambiente capturada por uma câmera convencional.

O gráfico exibido na Figura 29 mostra o comportamento do filtro com a configuração alternativa. Nota-se que, com a diferenciação dos valores de pose absolutos do LiDAR em valores relativos, a pose estimada final recebe uma influência maior dos dados obtidos pela odometria via *encoders* e pela odometria visual. Contudo, as covariâncias das poses absolutas obtidas pelo filtro crescem sem limites neste caso pela falta de dados absolutos disponíveis para limitá-las.

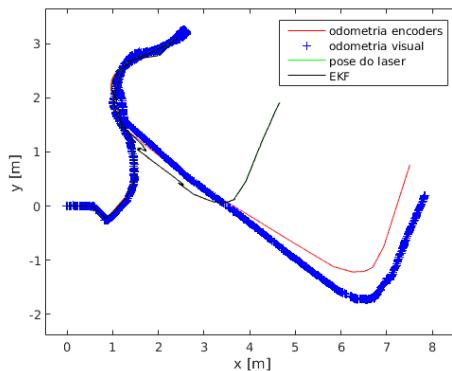


Figura 28 – Gráfico da trajetória (x,y) do robô vista de cima com a pose via laser reportando valores errados. As legendas se encontram na figura.

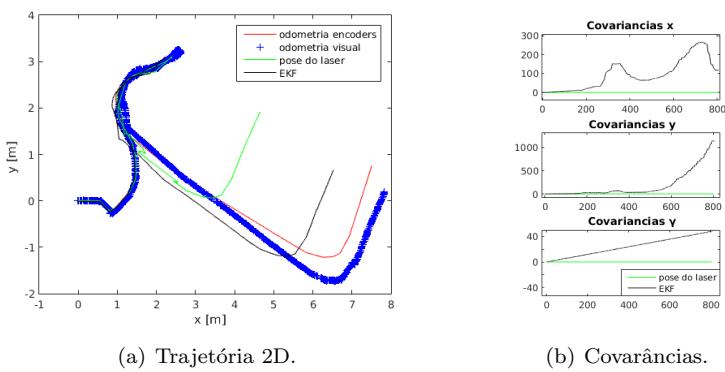


Figura 29 – Resposta do sistema com a configuração alternativa.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÃO

Neste trabalho foi desenvolvido um sistema de localização e mapeamento 3D, usando um EKF para realizar a fusão de 3 fontes de sensoriamento distintas. Os resultados mostrados na seção anterior comprovam a eficiência do sistema, sendo a estimativa dos estados realizada pelo filtro adequada e a fusão dos sensores considerando suas covariâncias um sucesso. Em pequenos testes, também foi observado que quando a odometria visual “se perde”, por exemplo em situações em que o robô se encontra de frente para um parede lisa com poucas características, as covariâncias dessa medição têm um valor bem alto (10^4), fazendo com que o filtro ignore essas medições. Outro fator importante desse sistema é a maior robustez na localização, pois se ao menos um sensor estiver funcionando, o EKF ainda consegue estimar a localização do robô. Por tanto, como os sensores são independentes, esse sistema funciona mesmo quando há falha de um ou dois sensores.

Desta forma, a utilização de técnicas probabilistas de localização em SLAM 3D na forma proposta por este trabalho demonstrou-se uma alternativa viável, resolvendo estes problemas de forma robusta e confiável, sendo capaz de obter a localização do robô durante toda sua trajetória.

O mapa obtido através de representação por nuvem de pontos pode, posteriormente, ser usado pelo robô para resolver o problema do robô sequestrado, em que o robô sabe exatamente onde está localizado, mas de repente ele é transferido para outro local sem estar ciente do ocorrido. Neste caso, o problema seria a detecção do sequestro e o descobrimento da nova localização. Neste projeto foi analisado o problema do robô despertado, no qual o robô é colocado em operação em uma posição arbitrária (no caso a origem do sistema de coordenadas).

Modelar um ambiente de forma tridimensional permite obter um alto grau de detalhamento, proporcionando uma compreensão muito mais abrangente do que mapas 2D. Esses modelos são de particular interesse para usuários remotos interessados no interior do ambiente que o robô percorre, como teleoperadores, que necessitam compreender o ambiente que seu robô vai operar, engenheiros e arquitetos, interessados em conhecer e manipular o interior de um ambiente, trabalhadores de resgate humano ou bombeiros, que gostariam de se familiarizar com um ambiente antes de entrar nele, ou mesmo de forma a conseguir

compreender um ambiente em que a presença humana é de difícil acesso ou impossível.

De modo geral, o sistema proposto neste projeto demonstrou-se capaz de resolver os problemas para os quais foi desenvolvido, permitindo a localização de um robô móvel durante sua trajetória e a construção e registro de um modelo 3D de seu ambiente de trabalho.

6.2 TRABALHOS FUTUROS

Em relação à trabalhos futuros, foram observados alguns aspectos e aplicações desse sistema, que podem ser explorados para expandir este trabalho:

- A localização do robô foi restringida no modo 2D e foi selecionado um ambiente plano, porém é possível a integração de outras variáveis de estado no filtro para uma localização 3D completa por meio de mais sensores, como uma unidade de medida inercial (IMU) ou GPS. Assim, o sistema poderia ser usado também em UAVs e veículos subaquáticos em ambientes irregulares, por exemplo.
- O sistema desenvolvido também pode ser testado em um ambiente maior ou ambientes interligados, como um edifício inteiro.
- O sistema proposto aqui poderia ser comparado com outra técnica de localização probabilística, como o UKF, fazendo uma análise comparativa entre resultados e tempo de processamento.
- O mapa obtido pode ser utilizado no robô para localização, planejamento de trajetórias, detecção de obstáculos, etc. Assim, seria possível integrar este sistema em um sistema maior, possivelmente autônomo.
- Todos os sensores utilizados neste projeto apresentam restrições e erros. A obtenção de valores bem precisos do deslocamento do robô (*ground truth*), como os obtidos por Sturm et al. (2012) através de um conjunto de câmeras de alta resolução no ambiente, possibilitaria uma melhor avaliação do sistema e um melhor refinamento nas configurações do EKF (principalmente na matriz de covariâncias de ruídos do processo \mathbf{Q}), para uma estimativa com erros menores.

REFERÊNCIAS

- ALAHI, A.; ORTIZ, R.; VANDERGHEYNST, P. Freak: Fast retina keypoint. In: **IEEE. Computer vision and pattern recognition (CVPR), 2012 IEEE conference on.** [S.l.], 2012. p. 510–517.
- ATIYA, S.; HAGER, G. D. Real-time vision-based robot localization. **IEEE Transactions on Robotics and Automation**, IEEE, v. 9, n. 6, p. 785–800, 1993.
- AULINAS, J. et al. The slam problem: a survey. **CCIA**, Citeseer, v. 184, n. 1, p. 363–371, 2008.
- BARBOSA, F. G. O. et al. **Sistema de localização, mapeamento e registro 3D para robótica móvel baseado em técnicas de visão computacional.** Dissertação (Mestrado) — Departamento de Automação e Sistemas, UFSC, 2017.
- BASDOGAN, C.; OZTIRELI, A. C. A new feature-based method for robust and efficient rigid-body registration of overlapping point clouds. **The Visual Computer**, Springer, v. 24, n. 7-9, p. 679–688, 2008.
- BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. **European conference on computer vision.** [S.l.], 2006. p. 404–417.
- BEKELE, D.; TEUTSCH, M.; SCHUCHERT, T. Evaluation of binary keypoint descriptors. In: **IEEE. Image Processing (ICIP), 2013 20th IEEE International Conference on.** [S.l.], 2013. p. 3652–3656.
- BELONGIE, S.; MALIK, J.; PUZICHA, J. Shape matching and object recognition using shape contexts. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 4, p. 509–522, 2002.
- BICCHI, A. et al. On the problem of simultaneous localization, map building, and servoing of autonomous vehicles. In: **Advances in control of articulated and mobile robots.** [S.l.]: Springer, 2004. p. 223–242.

BLACKWELL, D. Conditional expectation and unbiased sequential estimation. **The Annals of Mathematical Statistics**, JSTOR, p. 105–110, 1947.

BLAKE, A.; ISARD, M. The condensation algorithm-conditional density propagation and applications to visual tracking. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1997. p. 361–367.

BONIN-FONT, F.; ORTIZ, A.; OLIVER, G. Visual navigation for mobile robots: A survey. **Journal of intelligent and robotic systems**, Kluwer Academic Publishers, v. 53, n. 3, p. 263–296, 2008.

CALONDER, M. et al. Brief: Binary robust independent elementary features. In: SPRINGER. **European conference on computer vision**. [S.l.], 2010. p. 778–792.

COX, I. J.; LEONARD, J. J. Modeling a dynamic environment using a bayesian multiple hypothesis approach. **Artificial Intelligence**, Elsevier, v. 66, n. 2, p. 311–344, 1994.

CRUZ, L.; LUCIO, D.; VELHO, L. Kinect and rgbd images: Challenges and applications. In: IEEE. **Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on**. [S.l.], 2012. p. 36–49.

DELLAERT, F. et al. Monte carlo localization for mobile robots. In: IEEE. **Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on**. [S.l.], 1999. v. 2, p. 1322–1328.

DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: A survey. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 2, p. 237–267, 2002.

ENDRES, F. et al. 3-d mapping with an rgb-d camera. **IEEE Transactions on Robotics**, IEEE, v. 30, n. 1, p. 177–187, 2014.

FAN, B.; WU, F.; HU, Z. Rotationally invariant descriptors using intensity order pooling. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 34, n. 10, p. 2031–2045, 2012.

FILLIAT, D.; MEYER, J.-A. Map-based navigation in mobile robots:: I. a review of localization strategies. **Cognitive Systems Research**, Elsevier, v. 4, n. 4, p. 243–282, 2003.

- FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: **Readings in computer vision.** [S.l.]: Elsevier, 1987. p. 726–740.
- FOLETO, T. d. C. et al. Proposta de estimador não linear intermitente para sistemas de controle via rede sem fio. 2013.
- FREEMAN, W. T.; ADELSON, E. H. et al. The design and use of steerable filters. **IEEE Transactions on Pattern analysis and machine intelligence**, v. 13, n. 9, p. 891–906, 1991.
- GASPAR, J. **Visão para robótica móvel: Detecção de obstáculos sobre pavimento plano.** Tese (Doutorado) — Master thesis in Engenharia Electrotécnica e de Computadores, IST, 1994.
- GOOL, L. V.; MOONS, T.; UNGUREANU, D. Affine/photometric invariants for planar intensity patterns. In: SPRINGER. **European Conference on Computer Vision.** [S.l.], 1996. p. 642–651.
- HADDA, I.; KNANI, J. Global mapping and localization for mobile robots using stereo vision. In: IEEE. **Systems, Signals & Devices (SSD), 2013 10th International Multi-Conference on.** [S.l.], 2013. p. 1–6.
- HENRY, P. et al. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: CITESEER. **In the 12th International Symposium on Experimental Robotics (ISER.** [S.l.], 2010.
- HENRY, P. et al. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 31, n. 5, p. 647–663, 2012.
- HERVIER, T.; BONNABEL, S.; GOULETTE, F. Accurate 3d maps from depth images and motion sensors via nonlinear kalman filtering. In: IEEE. **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.** [S.l.], 2012. p. 5291–5297.
- HÖGMAN, V. **Building a 3D Map from RGB-D Sensors.** 2012.
- HOLTKAMP, M. J.; JONG, S. de. Robot localisation using sift and active monocular vision. 2006.

- HORN, B. K. Closed-form solution of absolute orientation using unit quaternions. **JOSA A**, Optical Society of America, v. 4, n. 4, p. 629–642, 1987.
- HUANG, A. S. et al. Visual odometry and mapping for autonomous flight using an rgb-d camera. In: **Robotics Research**. [S.l.]: Springer, 2017. p. 235–252.
- JOLLIFFE, I. T. Principal component analysis and factor analysis. In: **Principal component analysis**. [S.l.]: Springer, 1986. p. 115–128.
- KAMILA, N. K. **Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing**. [S.l.]: IGI Global, 2015.
- KE, Y.; SUKTHANKAR, R. Pca-sift: A more distinctive representation for local image descriptors. In: **IEEE. Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on**. [S.l.], 2004. v. 2, p. II–II.
- KHOSHELHAM, K.; ELBERINK, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. **Sensors**, Molecular Diversity Preservation International, v. 12, n. 2, p. 1437–1454, 2012.
- KOENDERINK, J. J.; DOORN, A. J. van. Representation of local geometry in the visual system. **Biological cybernetics**, Springer, v. 55, n. 6, p. 367–375, 1987.
- KROTKOV, E. Mobile robot localization using a single image. In: **IEEE. Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on**. [S.l.], 1989. p. 978–983.
- KULIS, B.; GRAUMAN, K. Kernelized locality-sensitive hashing for scalable image search. In: **IEEE. Computer Vision, 2009 IEEE 12th International Conference on**. [S.l.], 2009. p. 2130–2137.
- LABBE, M.; MICHAUD, F. Appearance-based loop closure detection for online large-scale and long-term operation. **IEEE Transactions on Robotics**, IEEE, v. 29, n. 3, p. 734–745, 2013.
- LABBÉ, M.; MICHAUD, F. Online global loop closure detection for large-scale multi-session graph-based slam. In: **IEEE. Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on**. [S.l.], 2014. p. 2661–2666.

- LABBÉ, M.; MICHAUD, F. Long-term online multi-session graph-based slam with memory management. **Autonomous Robots**, Springer, p. 1–18, 2017.
- LAZEBNIK, S.; SCHMID, C.; PONCE, J. Sparse texture representations using affine-invariant neighborhoods. In: CITESEER. In Proc. IEEE Conf. Comp. Vision Patt. Recog. [S.l.], 2003.
- LEUTENEGGER, S.; CHLI, M.; SIEGWART, R. Y. Brisk: Binary robust invariant scalable keypoints. In: IEEE. Computer Vision (ICCV), 2011 IEEE International Conference on. [S.l.], 2011. p. 2548–2555.
- LI, M.; YUAN, B. 2d-lda: A statistical linear discriminant analysis for image matrix. **Pattern Recognition Letters**, Elsevier, v. 26, n. 5, p. 527–532, 2005.
- LIN, R. et al. Vision-based mobile robot localization and mapping using the plot features. In: IEEE. Mechatronics and Automation (ICMA), 2012 International Conference on. [S.l.], 2012. p. 1921–1927.
- LING, L.; CHENG, E.; BURNETT, I. S. An iterated extended kalman filter for 3d mapping via kinect camera. In: IEEE. Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. [S.l.], 2013. p. 1773–1777.
- LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. Computer vision, 1999. The proceedings of the seventh IEEE international conference on. [S.l.], 1999. v. 2, p. 1150–1157.
- LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International journal of computer vision**, Springer, v. 60, n. 2, p. 91–110, 2004.
- LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous robots**, Springer, v. 4, n. 4, p. 333–349, 1997.
- MAIR, E. et al. Adaptive and generic corner detection based on the accelerated segment test. In: SPRINGER. European conference on Computer vision. [S.l.], 2010. p. 183–196.

METROPOLIS, N.; ULAM, S. The monte carlo method. **Journal of the American statistical association**, Taylor & Francis Group, v. 44, n. 247, p. 335–341, 1949.

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 27, n. 10, p. 1615–1630, 2005.

MIKSIK, O.; MIKOLAJCZYK, K. Evaluation of local detectors and descriptors for fast feature matching. In: **IEEE. Pattern Recognition (ICPR), 2012 21st International Conference on**. [S.l.], 2012. p. 2681–2684.

MONTEMERLO, M. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. **Aaai/iaai**, v. 593598, 2002.

MOORE, T.; STOUCH, D. A generalized extended kalman filter implementation for the robot operating system. In: **Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)**. [S.l.]: Springer, 2014.

MURPHY, K.; RUSSELL, S. Rao-blackwellised particle filtering for dynamic bayesian networks. In: **Sequential Monte Carlo methods in practice**. [S.l.]: Springer, 2001. p. 499–515.

NIN, M. C.; OSÓRIO, F. Navegação de robôs móveis autônomos e detecção de humanos baseada em sensor laser e câmera térmica. **Mostra Nacional de Robótica**, p. 1–6, 2011.

RAO, C. R. Information and the accuracy attainable in the estimation of statistical parameters. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 235–247.

ROSIN, P. L. Measuring corner properties. **Computer Vision and Image Understanding**, Elsevier, v. 73, n. 2, p. 291–307, 1999.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: SPRINGER. **European conference on computer vision**. [S.l.], 2006. p. 430–443.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: **IEEE. Computer Vision (ICCV), 2011 IEEE international conference on**. [S.l.], 2011. p. 2564–2571.

- RUSU, R. B. et al. Aligning point cloud views using persistent feature histograms. In: IEEE. **Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.** [S.l.], 2008. p. 3384–3391.
- RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). In: IEEE. **Robotics and automation (ICRA), 2011 IEEE International Conference on.** [S.l.], 2011. p. 1–4.
- SACCHETIN, M. C. **Análise e implementação de algoritmos para localização e mapeamento de robôs móveis baseada em computação reconfigurável.** Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, USP, 2006.
- SANDE, K. V. D.; GEVERS, T.; SNOEK, C. Evaluating color descriptors for object and scene recognition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 9, p. 1582–1596, 2010.
- SCHAFFALITZKY, F.; ZISSELMAN, A. Multi-view matching for unordered image sets, or ?how do i organize my holiday snaps?? In: SPRINGER. **European conference on computer vision.** [S.l.], 2002. p. 414–431.
- SHARP, G. C.; LEE, S. W.; WEHE, D. K. Icp registration using invariant features. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 24, n. 1, p. 90–102, 2002.
- SHI, J.; TOMASI, C. **Good features to track.** [S.l.], 1993.
- SICILIANO, B.; KHATIB, O. **Springer handbook of robotics.** [S.l.]: Springer Science & Business Media, 2008.
- SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots.** [S.l.]: MIT press, 2011.
- SIMMONS, R.; KOENIG, S. Probabilistic robot navigation in partially observable environments. In: **IJCAI.** [S.l.: s.n.], 1995. v. 95, p. 1080–1087.
- STURM, J. et al. A benchmark for the evaluation of rgbd slam systems. In: IEEE. **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.** [S.l.], 2012. p. 573–580.

SUGIHARA, K. Some location problems for robot navigation using a single camera. **Computer vision, graphics, and image processing**, Elsevier, v. 42, n. 1, p. 112–129, 1988.

THRUN, S. Probabilistic algorithms in robotics. **Ai Magazine**, v. 21, n. 4, p. 93, 2000.

THRUN, S. et al. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In: **AAAI/IAAI**. [S.l.: s.n.], 1998. p. 989–995.

THRUN, S. et al. Robotic mapping: A survey. **Exploring artificial intelligence in the new millennium**, v. 1, p. 1–35, 2002.

TOMATIS, N. Hybrid, metric-topological, mobile robot navigation. EPFL, 2001.

TUYTELAARS, T.; MIKOLAJCZYK, K. et al. Local invariant feature detectors: a survey. **Foundations and trends® in computer graphics and vision**, Now Publishers, Inc., v. 3, n. 3, p. 177–280, 2008.

VASKEVICIUS, N. et al. Efficient representation in three-dimensional environment modeling for planetary robotic exploration. **Advanced Robotics**, Taylor & Francis, v. 24, n. 8-9, p. 1169–1197, 2010.

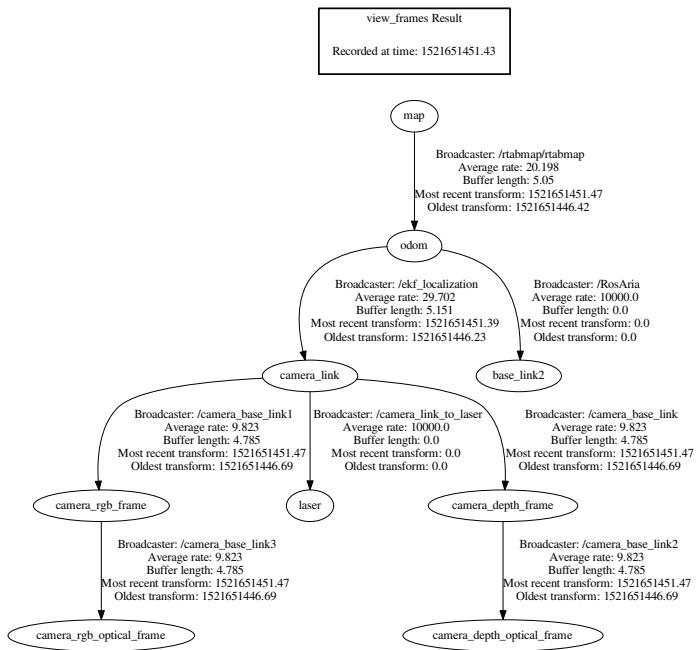
WANG, Z.; FAN, B.; WU, F. Local intensity order pattern for feature description. In: **IEEE. Computer Vision (ICCV), 2011 IEEE International Conference on**. [S.l.], 2011. p. 603–610.

ZHANG, Z. Iterative point matching for registration of free-form curves and surfaces. **International journal of computer vision**, Springer, v. 13, n. 2, p. 119–152, 1994.

ZITOVA, B.; FLUSSER, J. Image registration methods: a survey. **Image and vision computing**, Elsevier, v. 21, n. 11, p. 977–1000, 2003.

ZULIANI, M. Ransac for dummies. **Vision Research Lab, University of California, Santa Barbara**, Citeseer, 2009.

APÊNDICE A – Árvore de quadros

Figura 30 – Árvore de *frames*