UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Flávio Gabriel Oliveira Barbosa

SISTEMA DE LOCALIZAÇÃO, MAPEAMENTO E REGISTRO 3D PARA ROBÓTICA MÓVEL BASEADO EM TÉCNICAS DE VISÃO COMPUTACIONAL

Florianópolis

2017

Flávio Gabriel Oliveira Barbosa

SISTEMA DE LOCALIZAÇÃO, MAPEAMENTO E REGISTRO 3D PARA ROBÓTICA MÓVEL BASEADO EM TÉCNICAS DE VISÃO COMPUTACIONAL

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para obtenção do grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr.-Ing. Marcelo Ricardo Stemmer

Florianópolis

2017

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

Barbosa, Flávio Gabriel Oliveira
Sistema de localização, mapeamento e registro 3D para robótica móvel baseado em técnicas de visão computacional / Flávio Gabriel Oliveira Barbosa ; orientador, Marcelo Ricardo Stemmer - Florianópolis, SC, 2017.
138 p.

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

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Localização robótica. 3. Mapeamento robótico. 4. Reconstrução 3D. 5. Visão computacional. I. Stemmer, Marcelo Ricardo. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Automação e Sistemas. III. Título.

Flávio Gabriel Oliveira Barbosa

SISTEMA DE LOCALIZAÇÃO, MAPEAMENTO E REGISTRO 3D PARA ROBÓTICA MÓVEL BASEADO EM TÉCNICAS DE VISÃO COMPUTACIONAL

Esta Dissertação foi julgada e 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, 24 de Fevereiro de 2017.

Prof. Dr. Daniel Ferreira Coutinho
Coordenador do Curso

Prof. Dr.-Ing. Marcelo Ricardo Stemmer
Orientador

Banca Examinadora:

Prof. Dr. rer.nat. Aldo von Wangenheim

Prof. Dr. Mauricio Edgar Stivanello

Prof. Dr. Ubirajara Franco Moreno

À minha família.

AGRADECIMENTOS

A Deus, pela minha vida. Agradeço à minha mãe, Nilmar, pela motivação e confiança, ao meu pai, Jorge, pelo exemplo e amizade e à minha irmã, Luciana, pelo companheirismo e suporte. A todos os meus familiares, sobretudo ao meu avô Nilson, pelo incentivo. A todos os meus professores, em especial ao orientador, Prof. Dr. -Ing. Marcelo Ricardo Stemmer, pela minha formação.

Só levo a certeza de que muito pouco eu sei

Almir Sater/Renato Teixeira

RESUMO

A introdução de sistemas de visão computacional em robôs móveis se traduz em um significativo aumento de suas habilidades sensoriais, o que implica em uma maior versatilidade e segurança nas operações do robô.

Armazenar e manipular todas as imagens percebidas por um robô durante sua tarefa de localização e mapeamento visual é tipicamente intratável para cenários reais. A alternativa adotada por este trabalho é representar o ambiente de forma topológica, onde alguns quadros são selecionados, chamados *keyframes*, e representam locais visualmente distintos do ambiente. Assim, cada nó do mapa proposto corresponde a um quadro-chave, descrito por um conjuntos de características locais obtidas pelos descritores SIFT, SURF, ORB, BRIEF e BRISK. A seleção destes descritores baseou-se nas avaliações anteriores encontradas na literatura e em uma série de testes que verificaram habilidades importantes no contexto proposto.

Ao navegar em determinado ambiente, adquirir modelos 3D, que proporcionam uma compreensão muito mais abrangente do que mapas 2D, são de particular interesse para usuários remotos interessados no interior do ambiente que o robô percorre. O sistema proposto é baseado em registro de nuvens de pontos. Um Kinect acoplado ao robô captura imagens RGB e de profundidade, usadas para gerar nuvens de pontos que posteriormente são alinhadas na forma de registro, utilizando o alinhamento inicial SAC-IA com os descritores PFH e FPFH, e alinhadas através do algoritmo ICP.

As métricas de avaliação demonstraram que os sistemas propostos são capazes de localizar o robô com precisão, encontrando a localização global ao longo de toda a trajetória, sendo capaz de resolver os problemas do robô sequestrado e do robô despertado. O algoritmo de alinhamento mostrou bons resultados quanto a capacidade de criar modelos compreensíveis, porém demanda um alto custo computacional.

Desta forma, o presente trabalho propõe uma solução para que um robô acoplado com um Kinect percorra sua trajetória de forma autônoma, localizando-se e coletando nuvens de pontos que são usadas para criar um modelo 3D de seu ambiente de trabalho.

Palavras-chave: Localização robótica, mapeamento robótico, reconstrução 3D, visão computacional

ABSTRACT

The introduction of computer vision systems in mobile robots translates into a significant increase in their sensory habilities, which implies greater versatility and security in robot operations.
Storing and manipulating all the images perceived by a robot during its visual localization and mapping task is typically intractable for real scenarios. The alternative adopted by this work is to represent the environment in a topological form, where some frames are selected, called keyframes, and represent visually distinct locations from the environment. Thus, each proposed map node corresponds to a keyframe, described by a set of local characteristics obtained by SIFT, SURF, ORB, BRIEF and BRISK descriptors. The selection of these descriptors was based on previous evaluations found in the literature and on a series of tests that verified important skills in the proposed context.
When navigating through a particular environment, acquiring 3D models, which provide a much more comprehensive understanding than 2D maps, are of particular interest to remote users interested in the interior of the environment that the robot traverses. The proposed system is based on 3D point cloud registration. A Kinect is coupled to the robot and captures RGB and depth images, which are used to generate point clouds, aligned in the registration form, using the SAC-IA initial alignment with the descriptors PFH and FPFH, and fully aligned through the ICP algorithm.
The evaluation metrics demonstrated that the proposed systems are able to locate the robot with precision, being able to cope with the global localization problem throughout the whole trajectory, including the classic kidnapped robot and wake-up problems. The alignment algorithm showed good results in respect of creating comprehensive models, although it demands a high computational cost.
Therefore, the present work proposes a solution for a robot coupled with a Kinect to traverse its path autonomously, locating itself and collecting point clouds that are used to create a 3D model of its work environment.

Keywords: Robotic localization, robotic mapping, 3D reconstruction, computer vision

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

A robótica é uma ciência multidisciplinar em torno da qual outras áreas têm alcançado desenvolvimentos específicos expressivos (GASPAR, 1994). O crescimento e diversidade das suas aplicações, bem como o crescente interesse de diversos grupos de pesquisa, demonstram cada vez uma maior importância desta área em domínios como os dos sistemas produtivos, dos serviços e do lazer (HOLTKAMP; JONG, 2006).

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 povoado por objetos (SACCHETIN, 2005).

Segundo Gaspar (1994), um robô móvel consiste numa plataforma móvel sobre a qual é integrada percepção e ação sobre o mundo que o rodeia. A ação guia e controla 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 propõe ações. A utilização de imagens para percepção, por robôs móveis, do mundo a sua volta tem sido um desafio para as equipes de pesquisa que desenvolvem atividades na área da visão robótica.

Aplicações do mundo real exigem robôs móveis autônomos e modernos, que incluam uma variedade de sensores e uma alta capacidade computacional a fim de proporcionar essa autonomia. Meikle e Yates (1998) afirmam que na área da robótica o objetivo final é a construção de um robô completamente autônomo, que possa localizar-se e agir por conta própria.

Um requisito básico para qualquer entidade autônoma é a capacidade de aprender e explorar um ambiente desconhecido, de modo a ser capaz de encontrar o necessário para sustentar sua existência. Um robô móvel verdadeiramente autônomo (AMR - *Autonomous Mobile Robot*) precisa ser capaz de reabastecer ou recarregar-se, conforme necessário, além de ser capaz de identificar áreas ou situações perigosas (MEIKLE; YATES, 1998).

Através da visão podemos obter o conhecimento do ambiente e interagir inteligentemente com o nosso meio. Da mesma forma, robôs móveis podem tirar proveito dos recursos visuais que podem ser facilmente integrados com plataformas robóticas multissensoriais. Assim, eles representam uma resposta potencial à necessidade de capacidade de percepção para veículos autônomos.

Quando comparadas a outras técnicas, abordagens baseadas em

visão para navegação continuam a exigir muita atenção da comunidade científica voltada para a robótica móvel devido à sua capacidade de obter informações detalhadas sobre o ambiente, que podem não estar disponíveis através de combinações de outros tipos de sensores.

Segundo Leonard e Durrant-Whyte (1991), para um robô móvel autônomo, a navegação é a tarefa de mover-se com segurança de um local para outro. O problema geral de navegação pode ser formulado em quatro perguntas tendo o robô como centro de referência:

- Onde estou? O robô deve saber onde ele está localizado a fim de tomar decisões úteis. Este problema é chamado localização.

- Como é o mundo a minha volta? Na maioria das aplicações, o robô deve possuir ou construir uma representação de seu ambiente. Esta tarefa é conhecida como mapeamento.

- Para onde vou? A fim de cumprir alguma tarefa, o robô tem que saber para onde está indo. Trata-se de identificar um objetivo e este problema é, portanto, conhecido como reconhecimento de meta.

- Como eu chego lá? Uma vez que o robô sabe onde está e para onde tem que ir, ele deve decidir sobre como chegar lá. Encontrar uma maneira de chegar à meta é conhecido como planejamento de trajetória.

Em aplicações para robôs móveis, localização e mapeamento são habilidades fundamentais e importantes. Localizar-se e construir um mapa do ambiente permite que o robô possa alcançar outras metas. A localização é uma das características mais importantes para a navegação autônoma de robôs móveis, logo, robustez e precisão são caraterísticas fundamentais. Em geral, 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, conhecido como localização e mapeamento simultâneos (SLAM) (LIN et al., 2012).

Se o robô não sabe onde está localizado, pode ser difícil saber o que fazer a partir de então. Consequentemente, encontrar uma solução robusta e confiável é fundamental para responder às outras questões remanescentes. Além disso, essa é uma habilidade pré-requisito para a maioria dos sistemas de localização e mapeamento simultâneos e para navegação visual.

O problema de localização robótica pode ser tratado como local ou global. Em técnicas locais, a localização inicial do robô é conhecida pelo menos aproximadamente, e o objetivo é rastrear a localização enquanto o robô se move, compensando erros de outros sensores durante a navegação. Normalmente, se a posição for perdida, não pode ser recuperada (SE; LOWE; LITTLE, 2001).

Técnicas globais podem localizar o robô sem qualquer conhecimento prévio sobre a sua posição e são mais robustas que as locais, podendo lidar com situações em que o robô experimenta graves erros de localização e com o problema do robô sequestrado, em que além de não ser conhecida a pose inicial do robô, este pode ser retirado de sua posição durante o percurso e transferido para uma outra posição aleatória desconhecida (SACCHETIN, 2005) (FOX; BURGARD; THRUN, 1999).

Uma variedade de sensores têm sido utilizados para realizar a tarefa de localização (BICCHI et al., 2004), como o sistema de posicionamento global (GPS) (AGRAWAL; KONOLIGE, 2006) (REINA et al., 2007), *laser range finder* (LRF) (ZHANG; GHOSH, 2000) (SURMANN; NÜCHTER; HERTZBERG, 2003), sensores ultrassônicos (MORENO et al., 2002) e unidade de medida inercial (IMU) (YI et al., 2007). No entanto, são geralmente caros e podem ter restrições, como o grau de detalhe da informação ou velocidade limitada.

Outro problema de grande importância na busca da construção de robôs móveis verdadeiramente autônomos é o mapeamento, com aplicações em navegação, manipulação, mapeamento semântico, e telepresença (HENRY et al., 2012). Mapeamento robótico aborda o problema de aquisição de modelos espaciais dos ambientes físicos através de robôs móveis (THRUN et al., 2002).

Para construir uma representação do ambiente, os robôs devem possuir sensores que lhe permitam perceber o mundo exterior. No entanto, estes sensores estão sujeitos a erros, muitas vezes referidos como ruído de medição. Além disso, a maioria dos sensores do robô 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 LRF em aplicações tais como mapeamento ou representação 3D de interiores, vigilância, robótica e forense (KHOSHELHAM; ELBERINK, 2012). Sensores RGB-D, que capturam a imagem RGB juntamente com informação de profundidade por pixel, tais como o Kinect desenvolvido por Prime-Sense e Microsoft, proporcionam informação 3D a um preço acessível (HÖGMAN, 2012). O Kinect foi projetado principalmente para interação natural em um ambiente de jogo de computador. No entanto, as características dos dados capturados têm atraído a atenção de pesquisadores de outras áreas, incluindo mapeamento e modelagem 3D (KHOSHELHAM; ELBERINK, 2012). O processo de captação consiste em obter uma imagem colorida (RGB) e realizar uma medição de profundidade (D) com uma técnica conhecida como luz estruturada (CRUZ; LUCIO; VELHO, 2012).

A navegação robótica baseada em visão tem sido um grande objetivo para pesquisadores tanto da área de robótica como na área de visão computacional (NIN; OSÓRIO, 2011). Enquanto este problema possui diversas soluções para robôs equipados com dispositivos de determinação de distância, por uma variedade de razões, a tarefa continua a ser um desafio para robôs equipados apenas com sensores de visão.

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 e binoculares (estéreo), embora também existam sistemas baseados em configurações trinoculares. Outra estrutura que está ganhando popularidade por causa de suas vantagens é a de câmeras omnidirecionais, que possuem uma visão 360°do ambiente, e são normalmente obtidas combinando uma câmera convencional com um espelho convexo cônico, esférico, parabólico ou hiperbólico. Com este tipo de câmeras 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 num robô móvel traduz-se num significativo aumento das suas capacidades sensoriais e, portanto, num correspondente aumento de versatilidade e segurança na operação do robô. Espera-se, por exemplo, da visão robótica soluções ou simplificações para problemas de detecção e localização de objetos ou obstáculos, úteis no desempenho de tarefas de manipulação ou navegação.

Por outro lado, 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. Portanto, o tempo necessário para compreender e converter essa informação em dados úteis pode ser elevado (HADDA; KNANI, 2013).

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivo geral

O objetivo deste trabalho é a investigação e desenvolvimento de um método de localização, mapeamento visual baseado em aparência e reconstrução tridimensional de ambientes para robôs móveis em espaços internos.

1.1.2 Objetivos específicos

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

- Investigar implementações existentes e artigos;
- Implementar sistema de localização e mapeamento baseado em descritores de características locais;
- Implementar reconstrução 3D baseada em nuvem de pontos na forma de registro;
- Avaliar os sistemas propostos com base em critérios de avaliação a serem estudados.

1.2 ANÁLISE DE REQUISITOS

1.2.1 Sistema de localização e mapeamento

Os requisitos do sistema foram levantados a partir da necessidades encontradas para resolver o problema de localização na trajetória do robô ActivMedia Pioneer 3. Os requisitos funcionais e não-funcionais são descritos a seguir, assim como a forma de atuação no sistema desenvolvido.

1.2.1.1 Requisitos funcionais

RF-1. Escopo do sistema de localização e mapeamento

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 comunicar entre si, visto que o mapa proposto deve ser usado para inferir a localização, que por sua vez é feita com base no mapa.

RF-2. Composição 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.

RF-3. Compatibilidade do sistema de localização com sistemas robóticos

O sistema de localização e mapeamento deve suportar algum tipo de conexão com os demais sistemas responsáveis por outras tarefas do robô, como planejamento de trajetória e leituras de outros sensores. Isso implica em escolhas de linguagens de programação e de bibliotecas capazes de interagir com sistemas robóticos.

1.2.1.2 Requisitos não-funcionais

RNF-1. Tempo de setup

O tempo de setup da câmera do robô deve ser respeitado para início da tarefa de localização.

RNF-2. Tempo de correspondência

O tempo de correspondência entre a última imagem adquirida e a imagem a ser usada para inferir a localização do robô deve ser suficientemente pequeno para que a localização não ocorra após o robô ter percorrido considerável distância. Isso significa que a localização referente a imagem atual deve ser obtida antes que o robô percorra vários metros, estando em outra região ou outro contexto.

RNF-3. Marcadores naturais

O sistema deve ser capaz de se localizar sem o auxílio de marcadores artificiais, utilizando somente marcações naturais encontradas na trajetória do robô.

RNF-4. 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 sequestrado e do robô despertado.

1.2.2 Sistema de reconstrução 3D

Os requisitos do sistema foram levantados a partir da necessidades encontradas analisando-se o ambiente de trabalho do robô Pioneer e as limitações do dispositivo Microsoft Kinect. Os requisitos funcionais e não-funcionais são descritos a seguir, assim como a forma de atuação no sistema desenvolvido.

1.2.2.1 Requisitos funcionais

RF-1. Escopo do sistema de reconstrução
A meta do sistema de registro é utilizar as imagens da trajetória do robô para construir um modelo global tridimensional do ambiente de trabalho.
RF-2. Ambiente de trabalho
Os testes devem ser realizados em um Microsoft Kinect acoplado ao robô Pioneer em seu ambiente de trabalho.
RF-3. Compatibilidade do sistema de registro com sistemas robóticos
O sistema de reconstrução 3D deve suportar algum tipo de conexão com os demais sistemas responsáveis por outras tarefas do robô, como planejamento de trajetória e leituras de outros sensores. Isso implica em escolhas de linguagens de programação e de bibliotecas capazes de interagir com sistemas robóticos.

1.2.2.2 Requisitos não-funcionais

RNF-1. Características do Kinect
O sensor Kinect possui limitações em sua captura:

- Profundidade: cerca de 50 cm a 5 m. Embora seja possível aproximar-se (cerca de 40 cm), não é possível obter um visão completa a essa distância.

- Resolução Horizontal: 640×480 e 45°de FOV (campo de visão) vertical e 58°de FOV horizontal.

Portanto, o ambiente a ser reconstruído deve ser compatível com as limitações de captura do Microsoft Kinect.

RNF-2. Velocidade de movimento do Kinect

Outro desafio nos dados de imagem que os usuários devem ter em mente é que o Kinect usa um obturador rolante para a câmera colorida que pode levar a distorções de imagem quando a câmera é movida rapidamente (STURM et al., 2012). Além disso, o movimento do Kinect deve ser compatível com a taxa de captura, grandes e rápidas mudanças de direções podem causar nuvens de pontos sequenciais distantes entre si, o que dificulta o processo de registro. Assim, rápidos e bruscos movimentos devem ser evitados.

RNF-3. Confiabilidade da reconstrução

Deve haver uma certa verossimilhança entre realidade e reconstrução, de forma que o modelo final permita compreensão sobre a cena que representa a área de trabalho do robô.

RNF-4. Informações

O sistema proposto deve disponibilizar informações do ambiente que possam ser úteis para entender a área de trabalho do robô, como profundidade e cor.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

A fundamentação teórica necessária para o desenvolvimento deste trabalho é exposta no capítulo dois.

O capítulo três apresenta uma revisão bibliográfica, destacando o avanço da áreas de pesquisa em que esse trabalho se encontra.

A construção dos sistemas de localização, mapeamento e reconstrução para robótica móvel, incluindo as respectivas características, recursos e estruturas de funcionamento, é apresentada no capítulo quatro.

O capítulo cinco exibe a análise dos resultados dos experimentos 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 do trabalho desenvolvido e contínuo aprimoramento dos sistemas desenvolvidos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos que formam a base teórica deste trabalho. Serão expostas definições relacionadas à robótica móvel e visão computacional, abordando temas como a representação de posição e orientação de robôs móveis, de imagens por detectores e descritores de características locais, conceitos de localização e mapeamento para robôs móveis, além de nuvens de pontos, registro e a teoria necessária para realizar esta operação.

2.1 CONCEITOS RELACIONADOS À ROBÓTICA MÓVEL

A robótica alcançou um grande sucesso atualmente principalmente na produção industrial, como exemplificado pela Figura 1. No entanto, apesar dos sucessos, estes robôs comerciais sofrem de uma desvantagem fundamental: a falta de mobilidade. Um manipulador fixo tem uma gama limitada de movimentos que depende de onde ele é fixado. Em contraste, robôs móveis podem ser capazes de viajar por toda a fábrica, aplicando de forma flexível as suas habilidades onde quer que sejam mais eficazes (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004). Além disso, eles podem percorrer ambientes hostis, perigosos e inóspitos, como os robôs Mars Trigger no planeta Marte e Pioneer na usina nuclear de Chernobil após acidente nuclear, onde a presença de um ser humano pode ser de difícil acesso ou até mesmo impossível.

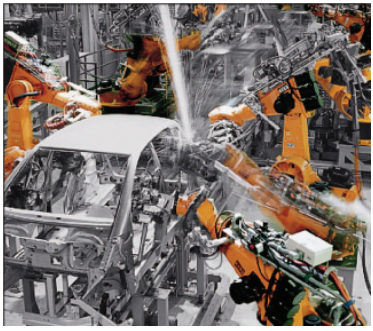


Figura 1 – Robôs para soldagem por ponto de alta velocidade (HIRZINGER et al., 2005)

Um robô móvel necessita de mecanismos de locomoção que o permitam movimentar-se ao longo do seu ambiente. Siegwart, Nourbakhsh e Scaramuzza (2004) afirmam que a navegação é uma das competências mais difíceis exigidas de um robô móvel. O sucesso nesta habilidade requer êxito nos quatro blocos que compõem a navegação: a percepção, o robô deve interpretar seus sensores para extrair dados significativos; localização, o robô tem de determinar a sua posição no ambiente (Figura 2); cognição, o robô deve decidir como agir para atingir os seus objetivos; e controle de movimento, o robô deve modular as suas saídas do motor para atingir a trajetória desejada.

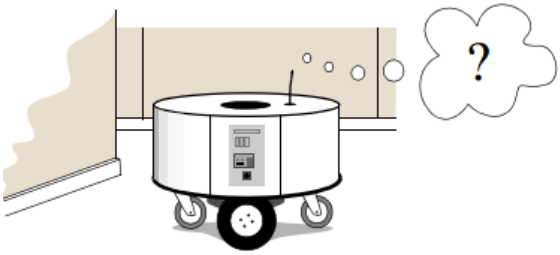


Figura 2 – Onde estou? (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004)

2.1.1 Representação de posição e orientação

A menos que expressamente indicado o contrário, os mecanismos robóticos são sistemas de corpos rígidos conectados por articulações. A posição e orientação de um corpo rígido no espaço são denominados coletivamente a pose. 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). Entender o comportamento mecânico do robô é necessário tanto para projetar robôs móveis apropriados para determinadas tarefas como para criar um algoritmo de controle para uma instância de hardware do robô móvel (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004).

Siciliano e Khatib (2008) afirmam que a cinemática espacial e de corpo rígido pode ser vista como um estudo comparativo de diferentes maneiras de representar a pose de um corpo. Translações e rotações,

referidas em combinação como deslocamentos de corpo rígido, também são expressas com essas representações.

O espaço de trabalho de um robô manipulador é crucial porque define a gama de posições possíveis que podem ser alcançadas pelo seu atuador final em relação ao seu ambiente. O espaço de trabalho de um robô móvel é igualmente importante porque define a gama de poses possíveis que ele pode alcançar em seu ambiente (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004).

Um sistema de coordenadas i consiste de uma origem, denotada O_i, e uma tríade 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 algum outro, de modo que possa ser expresso como a pose de uma coordenada relativa a outra. Do mesmo modo, 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 numa posição fixa no interior do sistema de referência fixo, conforme apresentado pela Figura 3 (SICILIANO; KHATIB, 2008).

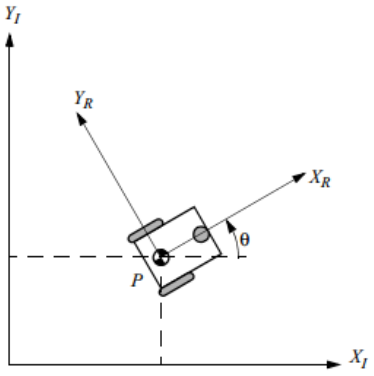


Figura 3 – Os sistemas de referência global e local do robô em 2D (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004).

2.1.2 Posição e deslocamento

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 da origem do sistema de coordenadas i relativa ao sistema de coordenadas j pode ser indicada pelo seguinte vetor 3×1:

$$ {}^{j}p_i = \begin{pmatrix} {}^{j}p_i^x \\ {}^{j}p_i^y \\ {}^{j}p_i^z \end{pmatrix} \tag{2.1} $$

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 na 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 de suas posições antes e depois da translação. Por outro lado, a posição de um corpo pode ser representada como uma translação que leva o corpo de uma posição na qual o sistema de coordenadas fixo ao corpo coincide com o sistema de coordenadas fixo para a posição atual na qual os dois sistemas não coincidem. Assim, qualquer representação de posição pode ser usada para criar uma representação de deslocamento, e vice-versa (SICILIANO; KHATIB, 2008).

Na Figura 4, o sistema de coordenadas x_j, y_j e z_j é deslocado do sistema de coordenadas x_i, y_i e z_i na direção do vetor unitário $\vec{e_j}$ na distância q_j. O correspondente par de eixos de ambos sistemas de coordenadas são paralelos. O corpo é transladado e todos os seus pontos são deslocados na mesma direção e pela mesma distância. A translação entre sistemas de coordenadas é determinada por um vetor de translação $q_j\vec{e}_j$. A posição do ponto A é dada pelo vetor $\vec{a}$ e não é alterada com respeito ao sistema de coordenadas x_j, y_j, z_j que está ligado ao corpo. A posição do ponto A em relação ao sistema de coordenadas x_i, y_i e z_i é dependente do comprimento e da direção do vetor $\vec{a'}$ (LENARCIC; BAJD; STANIŠIĆ, 2012).

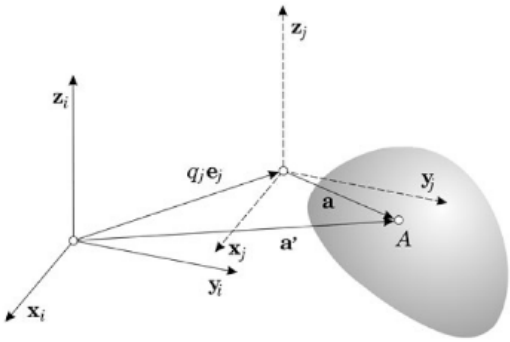


Figura 4 – Vetor de translação $q_j\vec{e}_j$ (LENARCIC; BAJD; STANIŠIĆ, 2012)

2.1.3 Orientação

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).

Há uma gama significativamente maior na representação da orientação do que na posição. Alguns exemplos são ângulos de Euler, ângulo/eixo, quatérnions e matrizes de rotação.

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

A representação da orientação do robô por quatérnions é extremamente útil para problemas em robótica que resultam em singularidades representacionais na notação vetor/matriz, como os apresentados por ângulos de Euler (SICILIANO; KHATIB, 2008).

Horn (1987) afirma que um quatérnion q, por definição, tem a forma:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2.2}$$

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, comumente referidos como parâmetros de Euler, e i, j e k são operadores. Os operadores são definidos para satisfazer as seguintes regras combinatórias:

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

Muitas vezes, q_0 é referido como a parte escalar do quatérnion, e $(q_1 \quad q_2 \quad q_3)^T$ é referido como a parte vetorial.

Um quatérnion cuja norma é igual a 1 é chamado quatérnion unitário. 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 $p = (p_x \quad p_y \quad p_z)^T$ pode ser expresso como um quatérnion $p = p_x i + p_y j + p_z k$.

Rotações tridimensionais podem ser representadas por quatérnions unitários (KELLY, 2013):

$$\tilde{q} = cos(\frac{\theta}{2}) + \hat{w} sen(\frac{\theta}{2}) \tag{2.4}$$

Onde a equação 2.4 representa o operador que gira pelo ângulo θ ao redor do eixo cujo vetor unitário é $\hat{w}$.

O negativo $-\tilde{q}$ representa a mesma rotação (direção oposta ao vetor negativo) enquanto o conjugado $\tilde{q}*$ produz a rotação inversa (como o inverso de uma matriz de rotação).

Além disso, Siciliano e Khatib (2008) afirmam que os quatérnion 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 ângulo de rotação, enquanto q_1, q_2 e q_3 definem o eixo de rotação.

2.1.4 Navegação robótica

Bellotto et al. (2008) afirmam que a navegação pode ser descrita grosseiramente como o processo de determinar um caminho adequado e seguro entre um ponto de partida e um ponto de meta para um robô que viaja entre eles. Diferentes sensores foram utilizados para este fim, o que levou a um espectro variado de soluções. Em particular, nas últimas décadas, a navegação visual para robôs móveis tornou-se uma fonte de inúmeras contribuições de pesquisa, a partir dos domínios de visão e controle, uma vez que as estratégias de navegação baseadas em visão podem aumentar o escopo de aplicação de veículos móveis autônomos, se tornando cada vez mais comum em aplicações como localização, construção automática de mapas, navegação autônoma, seguimento de caminhos, inspeção, monitoramento ou detecção de situações de risco.

Em muitos casos, o desempenho de um bom algoritmo de navegação está profundamente associado a uma localização precisa do robô no ambiente. Assim, em robótica móvel, a localização desempenha um papel fundamental para a tarefa de navegação, uma vez que é necessária para praticamente todo o tipo de planejamento de trajetória. Para atingir uma meta, um robô móvel autônomo deve ser capaz de localizar-se dentro do ambiente onde está atuando e relativamente ao destino alvo (BELLOTTO et al., 2008).

A localização é o processo de inferir a pose do robô dentro de um mapa (FILLIAT; MEYER, 2003). O mapeamento robótico aborda o problema da aquisição de modelos espaciais de ambientes físicos através de robôs móveis e é geralmente considerado 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. 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 posição é limitada pela fidelidade do mapa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

As abordagens existentes podem ser agrupadas em três paradigmas distintos: topológico, métrico e híbrido. Abordagens no paradigma métrico capturam as propriedades geométricas do ambiente do robô. As abordagens no paradigma topológico, por outro lado, descrevem a conectividade de diferentes locais, geralmente representando ambientes como uma lista de lugares significativos conectados via arcos. O mo-

delo híbrido permite o uso das abordagens diferentes e complementares (THRUN et al., 1998) (TOMATIS, 2001).

Siegwart, Nourbakhsh e Scaramuzza (2004) afirmam que se alguém pudesse anexar um sensor de GPS (sistema de posicionamento global) preciso a um robô móvel, grande parte do problema de localização seria evitado. O GPS iria informar o robô de sua posição exata, dentro de casa e ao ar livre, de modo que a resposta à pergunta, "Onde estou?", estaria sempre 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, bem como robôs móveis em miniatura, como robôs de mesa e os nanorrobôs de navegação no corpo do futuro. Além disso, as tecnologias GPS podem não funcionar em ambientes fechados ou em áreas obstruídas e, portanto, são limitadas em seu espaço de trabalho.

Os sensores e atuadores do robô desempenham um papel integral em todas as formas de localização. É por causa da imprecisão e incompletude desses sensores e atuadores que a localização coloca-se como um desafio tão difícil. 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, 2004).

Dado um mapa e um local de meta, uma competência importante que robôs móveis devem ser capazes de resolver é o planejamento de trajetória. Essa tarefa envolve a identificação de uma trajetória que fará com que o robô alcance o local-objetivo quando executado. O planejamento do trajetória é uma competência estratégica de resolução de problemas, pois o robô deve decidir o que fazer em longo prazo para atingir seus objetivos (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004).

Siegwart, Nourbakhsh e Scaramuzza (2004) afirmam que uma segunda competência igualmente importante é evitar obstáculos. Dadas as leituras de sensores em tempo real, evitar obstáculos significa modular a trajetória do robô de forma a impedir colisões.

Assim, para atingir seu objetivo, o robô deve incorporar novas informações obtidas durante a execução do planejamento. À medida que o tempo avança, o ambiente muda e os sensores do robô coletam novas informações. É precisamente onde a reação se torna relevante. No

melhor dos casos, a reação irá modular o comportamento do robô localmente para corrigir a trajetória planejada para que o robô ainda atinja a meta. Às vezes, novas informações imprevistas exigirão mudanças nos planos estratégicos do robô e, assim, idealmente, o planejador também incorporará novas informações à medida que essas novas informações forem recebidas.

2.1.5 Navegação através de visão computacional em ambientes internos

O uso 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).

Segundo DeSouza e Kak (2002), há cerca de trinta anos atrás teria sido impossível para um robô móvel encontrar o seu caminho em um corredor tão confuso como o mostrado na Figura 5, mas atualmente não é um desafio.

Figura 5 – Robôs móveis podem agora localizar-se usando a visão computacional em cenas com este tipo de complexidade (DESOUZA; KAK, 2002).

Alguns dos primeiros sistemas de visão desenvolvidos para a robótica móvel dependiam fortemente da geometria do espaço e de outras informações métricas para conduzir os processos de visão e realizar auto-localização. Em particular, o espaço interior era representado por modelos CAD de complexidade variável. Esses modelos podem ser substituídos por outros mais simples, como mapas de ocupação, mapas topológicos ou mesmo sequências de imagens. Quando as sequências de imagens são usadas para representar o espaço, as imagens obtidas durante a navegação são submetidas a algum tipo de correspondência baseada na aparência entre a percepção (imagem real) e a expectativa (imagem objetivo ou imagens-meta armazenadas no banco de dados) (DESOUZA; KAK, 2002).

DeSouza e Kak (2002) classificam a navegação de robôs móveis em ambientes interiores em três grupos:

- **Navegação baseada em mapas**: Estes são sistemas que dependem de algum tipo de representação do ambiente previamente fornecida;
- **Navegação baseada em construção de mapas**: Estes são sistemas que usam sensores para construir seus próprios modelos geométricos ou topológicos do ambiente e usam então estes modelos para a navegação;
- **Navegação sem mapas**: Estes são sistemas que não usam nenhuma representação explícita sobre o espaço no qual a navegação deve ocorrer, mas sim recorrem ao reconhecimento de objetos encontrados no ambiente ou ao rastreamento desses objetos, gerando movimentos baseados em observações visuais.

Uma outra abordagem, proposta por Bonin-Font, Ortiz e Oliver (2008), argumenta que independentemente do tipo de veículo, os sistemas que usam a visão para a navegação podem ser divididos grosseiramente naqueles que precisam de conhecimento prévio de todo o ambiente e aqueles que percebem o ambiente à medida que navegam através dele. Os sistemas que precisam de um mapa, por sua vez, podem ser subdivididos quanto ao tipo de mapa que utilizam.

Em sistemas de navegação sem mapa, a navegação é alcançada sem qualquer descrição prévia do ambiente, geralmente, através de técnicas reativas em que os movimentos necessários do robô são determinados observando e extraindo a informação relevante sobre os elementos no ambiente. Estes elementos podem ser as paredes, objetos tais como mesas, entradas de portas, etc. Não é necessário que sejam conhecidas

posições absolutas (ou mesmo relativas) destes elementos do ambiente. No entanto, a navegação só pode ser realizada com relação a esses elementos (DESOUZA; KAK, 2002) (BONIN-FONT; ORTIZ; OLIVER, 2008).

De acordo com Bonin-Font, Ortiz e Oliver (2008), duas técnicas principais para esse tipo de navegação devem ser citadas: navegação por fluxo óptico e navegação baseada em aparência. As soluções baseadas em fluxo óptico estimam o movimento de objetos ou características dentro de uma sequência de imagens. As técnicas de correspondência baseadas em aparência são realizadas através do armazenamento de imagens em uma fase de gravação anterior. Essas imagens são usadas como modelos. O robô se auto-localiza e navega no ambiente correspondendo o quadro atual com os modelos armazenados.

Em abordagens baseadas em aparência, na maioria dos casos o robô só tem acesso a algumas sequências de imagens que o ajudam a chegar ao seu destino, ou algumas imagens predefinidas (imagens meta/alvo) que ele pode usar para rastrear e buscar. Em vez de usar abordagens baseadas em aparência para memorizar e buscar locais, uma abordagem de navegação simbólica também pode ser usada. Neste caso, o robô recebe comandos como "vá até a porta" ou "vá até a mesa à sua frente" etc. e usa as informações simbólicas contidas nesses comandos para estabelecer os pontos de referência que precisa reconhecer e o caminho que precisa tomar para alcançar a meta (DESOUZA; KAK, 2002).

A navegação baseada em mapas consiste em fornecer ao robô um modelo do ambiente. Estes modelos podem conter diferentes graus de detalhe, variando de um modelo CAD completo do ambiente a um simples grafo de interconexões ou inter-relações entre os elementos no ambiente. Em alguns dos primeiros sistemas de visão, o conhecimento do ambiente consistia em uma representação de grade em que cada objeto no ambiente era representado por uma projeção 2D do seu volume sobre o plano horizontal. Tal representação é geralmente referida como "mapa de ocupação" (DESOUZA; KAK, 2002) (BONIN-FONT; ORTIZ; OLIVER, 2008).

Uma vez que a ideia central em qualquer navegação baseada em mapas é fornecer ao robô, direta ou indiretamente, uma sequência de marcações esperadas para serem encontradas durante a navegação, a tarefa do sistema de visão é então procurar e identificar essas marcações observadas nas imagens. Uma vez identificadas, o robô pode usar o mapa fornecido para estimar sua posição (auto-localização), combinando a observação (imagem) com a expectativa (descrição das marcações na base de dados). Os cálculos envolvidos neste tipo de localização

baseada em visão podem ser divididos nas seguintes quatro etapas (DESOUZA; KAK, 2002):

- Adquirir informações sensoriais. Para a navegação baseada em visão, isso significa adquirir e digitalizar imagens da câmera;
- Detectar marcações. Normalmente, isso significa extrair bordas, suavizar, filtrar e segmentar regiões com base em diferenças nos níveis de cinza, cor, profundidade ou movimento. Outra técnica utiliza detectores e descritores de características locais;
- Estabelecer correspondências entre observação e expectativa. Nesta etapa, o sistema tenta identificar as marcações observadas pesquisando na base de dados possíveis correspondências de acordo com alguns critérios de medição;
- Calcular a posição. Uma vez obtida uma correspondência (ou um conjunto de correspondências), o sistema precisa calcular sua posição como uma função das marcações observadas e suas posições na base de dados.

Segundo Bonin-Font, Ortiz e Oliver (2008), para se auto-localizar, os algoritmos de rastreamento de referências determinam a posição do robô, detectam marcações na imagem da câmera e os acompanham nas cenas consecutivas. As marcações podem ser artificiais (Figura 6) ou naturais, tais como portas, janelas, etc. Em ambos os casos, o robô precisa conhecer a identidade dos pontos de referência para poder rastreá-los. Este método tem sido usado em sistemas de navegação baseados em mapas e em algumas arquiteturas de navegação reativa. Na maioria dos casos, o rastreamento de pontos de referência nesses sistemas é realizado usando a correspondência simples de modelos. As câmeras são geralmente montadas no robô para que eles olhem lateralmente para as paredes onde os marcos estão montados ou para baixo no chão onde uma fita especial pode ser colada. Quando as câmeras não são montadas lateralmente, as propriedades das formas usadas para as marcações artificiais permitem que algoritmos simples sejam usados para a localização do robô. Enquanto as câmeras montadas lateralmente simplificam o problema da detecção de pontos de referência (eliminando efeitos de escala e de perspectiva), elas também limitam a liberdade em relação ao local onde o robô pode se mover (DESOUZA; KAK, 2002).

DeSouza e Kak (2002) afirmam que os sistemas baseados em mapas podem ser classificados de acordo com o tipo de localização que buscam:

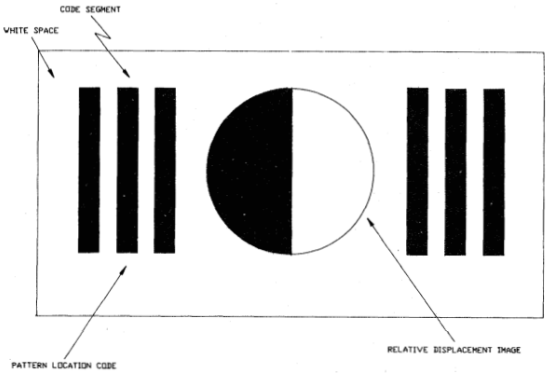


Figura 6 – Marcação artificial usada em (KABUKA; ARENAS, 1987).

- **Localização absoluta**: 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 (ISARD; BLAKE, 1998) (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), dentre outros;

- **Localização incremental**: Em um grande número de situações práticas, a posição inicial do robô é conhecida pelo menos aproximadamente. Em tais casos, o algoritmo de localização deve simplesmente acompanhar as incertezas na posição do robô quando executa comandos de movimento e, quando as incertezas excedem um limite, usar seus sensores para uma nova correção em sua posição. Em geral, as técnicas probabilísticas evoluíram como a abordagem preferida para a representação e a atualização das incertezas posicional à medida que o robô se move.

Essas abordagens de navegação baseadas em visão exigem que o

robô possua um mapa (ou um modelo) do ambiente. Mas essas descrições nem sempre são fáceis de gerar, especialmente se informações métricas são necessárias. Portanto, muitos pesquisadores propuseram robôs autônomos ou semiautônomos que poderiam explorar seu ambiente e construir uma representação interna dele durante a fase de navegação (DESOUZA; KAK, 2002). A construção de mapas e a autolocalização no ambiente de navegação são duas funcionalidades que os sistemas não reativos tendem a incorporar. Nas abordagens padrão de construção de mapas, assume-se que a localização no ambiente pode ser calculada por alguma outra técnica, enquanto que em abordagens de localização pura, o mapa do ambiente está presumivelmente disponível. Os robôs que usam esta abordagem de navegação precisam rastrear sua própria posição e orientação no ambiente de forma contínua (BONIN-FONT; ORTIZ; OLIVER, 2008).

A escolha do mapa tem impacto na localização e vice-versa. Bonin-Font, Ortiz e Oliver (2008) afirmam que mapas métricos incluem informações como distâncias ou tamanhos de células do mapa com respeito a um sistema de coordenadas predefinido e, em geral, também são mais sensíveis a erros de sensores. Mapas métricos precisos são essenciais para uma boa localização e uma localização precisa torna-se necessária para construir um mapa preciso. Se a exploração e mapeamento de um ambiente desconhecido é feita automaticamente e on-line, o robô deve realizar três tarefas: exploração/navegação segura, mapeamento e localização, de preferência de forma simultânea.

Uma outra estratégia são os mapas de ocupação. Uma grade de ocupação representa uma região observada e cada célula da grade é rotulada com a probabilidade de ser ocupada por um objeto. Embora as abordagens baseadas em grades de ocupação sejam capazes de gerar mapas ricos em detalhes geométricos, a extensão em que a geometria resultante pode ser confiada para a navegação subsequente depende naturalmente da precisão da odometria dos robôs e das incertezas dos sensores durante a construção do mapa. Além disso, para espaços em larga escala e complexos, as representações resultantes podem não ser computacionalmente eficientes para o planejamento do trajetória, localização, etc. Uma representação topológica do ambiente é uma alternativa a uma grade de ocupação. Esses sistemas baseiam-se na geração de um grafo de nós representando o espaço que podem armazenar informações métricas para cada nó reconhecido durante o processo de navegação. As diferentes abordagens diferem do que constitui um nó, como um nó pode ser distinguido de outros, a possibilidade de usar as incertezas do sensor ou como essas incertezas são computadas. Uma

das principais dificuldades é o reconhecimento de nós previamente visitados (em abordagens métricas, por outro lado, se a odometria e os sensores forem suficientemente precisos, as distâncias calculadas entre as diferentes características do espaço ajudam a estabelecer locais de identificação previamente visitados) (DESOUZA; KAK, 2002) (BONIN-FONT; ORTIZ; OLIVER, 2008).

Mapas topológicos são uma representação gráfica do ambiente. Cada nó corresponde a uma característica ou zona do ambiente e pode ser associado a uma ação, como girar, atravessar uma porta, parar ou seguir em frente. Geralmente, não há distâncias absolutas, nem referências a qualquer sistema de coordenadas para medir o espaço. Este tipo de mapas são adequados para navegação qualitativa de longa distância, e especialmente para o planejamento de trajetória. Em geral, não representam explicitamente o espaço livre para que os obstáculos sejam detectados e evitados on-line por outros meios. Os mapas topológicos são simples e compactos, ocupam menos memória do computador e, consequentemente, aceleram os processos computacionais de navegação (BONIN-FONT; ORTIZ; OLIVER, 2008).

2.1.6 Obtenção de modelos 3D de ambientes por robôs móveis

A crescente necessidade de rápida caracterização e quantificação de ambientes complexos criou desafios para a análise de dados. Esta necessidade crítica vem de muitas áreas importantes, incluindo automação industrial, arquitetura, agricultura e construção ou manutenção de túneis e minas. Por um lado, dados 3D precisos de ambientes são necessários para a concepção da fábrica, gestão de instalações, planejamento urbano e regional. As informações 3D disponíveis permitem que robôs autônomos naveguem em ambientes desconhecidos, por exemplo, no campo da robótica de inspeção e salvamento (SURMANN; NÜCHTER; HERTZBERG, 2003).

Apesar da robótica móvel caminhar para robôs completamente autônomos, grande parte dos robôs móveis ainda são teleoperados. Um problema disso é que muitas vezes esses robôs devem percorrer ambientes em que a presença humana não é viável e que muitas vezes não são conhecidos. Uma solução para este problema é que robôs possam construir um modelo 3D que possa auxiliar nas suas tarefas, como a de planejamento de trajetória, além de compreensão deste ambiente.

Liu et al. (2001) afirmam que um grande número de robôs móveis que navegam em ambientes internos baseiam-se em mapas de ambiente

para navegação. Quase todos os algoritmos existentes para a aquisição de tais mapas operam em 2D apesar do fato de que os ambientes de robôs serem tridimensionais. Normalmente se supõe que duas dimensões são suficientes, uma vez que o robô está confinado a um plano bidimensional. No entanto, modelar um ambiente em 3D tem duas vantagens importantes: Primeiro, mapas 3D facilitam a desambiguação de diferentes lugares, já que os modelos 3D são muito mais ricos que os modelos 2D e, portanto, possuem menos ambiguidades. Em segundo lugar, eles são de particular interesse se o objetivo do mapeamento vai além da navegação do robô. Modelos 3D são mais adequados para usuários remotos interessados no interior de um edifício, como arquitetos, trabalhadores de resgate humano ou bombeiros que gostariam de se familiarizar com um ambiente antes de entrar nele. Passar de 2D para 3D não é apenas uma extensão trivial

O problema da detecção e modelagem do ambiente é complexo, uma vez que uma série de questões científicas fundamentais estão envolvidas nesta pesquisa. Primeiro, o controle de um robô móvel autônomo e o ambiente de varredura com um sensor 3D. A segunda é como criar uma cena consistente volumétrica em um sistema de coordenadas comum a partir de múltiplas vistas. Terceiro é o cálculo dos pontos de vista seguintes para proporcionar cobertura completa e eficiente da cena e para eliminar a oclusão sob a restrição de minimizar o comprimento total do percurso entre estes pontos (SURMANN; NÜCHTER; HERTZBERG, 2003).

Técnicas de *Structure from Motion* (SfM) de visão computacional normalmente usam imagens de uma câmera em movimento. Ao extrair e combinar características visuais em várias visualizações, *bundle ajustment* (TRIGGS et al., 1999) pode ser usado para estimar tanto a postura da câmera quanto um modelo 3D esparso dos pontos de característica. Além disso, vários métodos têm sido propostos para calcular mapas densos de profundidade a partir de dados de imagem. Câmeras estéreo podem obter essa informação de profundidade através da geometria epipolar. Outra alternativa são sensores de profundidade como o Microsoft Kinect que podem ser integrados em plataformas robóticas abrindo assim novas possibilidades para abordagens em reconstrução 3D. Estes dados podem ser reprojetados como um conjunto de pontos discretos 3D (ou nuvem de pontos) (IZADI et al., 2011). Consequentemente, essas imagens de profundidade podem ser eficientemente registradas (alinhadas de forma a criar um modelo global) e integradas (ZHANG et al., 2012) (RUSU, 2010) (BASDOGAN; OZTIRELI, 2008).

2.2.1 Detectores e descritores de características locais

Uma alternativa para encontrar marcações naturais em um ambiente são descritores de características locais. Uma característica local é um padrão de imagem que difere da sua vizinhança imediata. Os detectores devem fornecer regiões que são utilizadas para calcular os descritores, extraindo as características da imagem. Uma vez que foram detectadas, estas características podem ser extraídas. O extrator calcula um descritor dos pixeis em torno de cada ponto de interesse (TUYTELAARS; MIKOLAJCZYK, 2007).

De acordo com Mikolajczyk et al. (2005), muitas técnicas diferentes para descrever regiões de interesse em imagem têm sido desenvolvidas. O descritor mais simples é um vetor de pixeis da imagem. Correlação cruzada pode então ser utilizada 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, técnicas baseadas em descritores de característica para encontrar correspondências entre as imagens são utilizadas com maior frequência.

A seguir serão apresentados alguns descritores de ampla utilização.

2.2.1.1 SIFT

O descritor SIFT(*Scale-invariant feature transform*) transforma uma imagem em uma coleção de vetores de características locais, sendo tolerante a translação, escala e rotação de imagens, e parcialmente invariante a mudanças de iluminação e projeções afim ou 3D (LOWE, 1999). O algoritmo pode ser sumarizado pelo fluxograma apresentado na Figura 7.

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).

Segundo Lowe (1999), 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, $I(x, y)$:

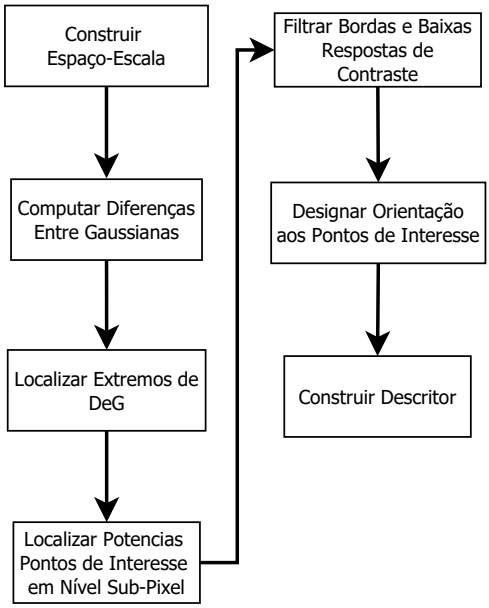


Figura 7 – Fluxograma de Funcionamento do SIFT. Adaptado de (CLEMONS, 2014).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.5}$$

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

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.6}$$

O resultado desta operação é uma pirâmide de imagens divida em oitavas, que se diferenciam por um fator de escala de dois, e escalas. 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:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.7}$$

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). Desta forma, *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 2×2 Hessiana (H) é usada para calcular a curvatura principal. Para o detector de cantos de Harris, para as bordas, um valor próprio é maior que o outro. Lowe (1999) usa uma função simples: se esta proporção é superior a um 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 pixeis (LOWE, 1999):

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{2.8}$$

$$\theta(x,y) = tan^{-}1(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))}) \tag{2.9}$$

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-espaço 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 $k \times k$ pixeis 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 pixeis. Assim, um total de 128 valores estão disponíveis. São representados como um vetor para formar o descritor do ponto de interesse (KAMILA, 2015).

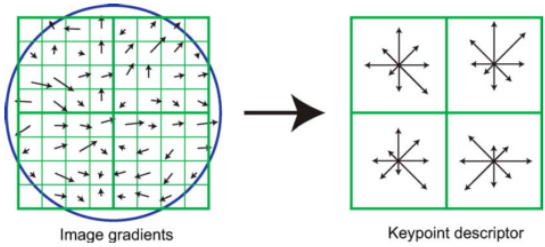


Figura 8 – Descritor SIFT (LOWE, 2004)

2.2.1.2 SURF

O descritor SURF(*Speeded-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).

Segundo 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):

$$H(x, \sigma) = \begin{bmatrix} Lxx(x, \sigma) & Lxy(x, \sigma) \\ Lxy(x, \sigma) & Lyy(x, \sigma) \end{bmatrix} \tag{2.10}$$

Onde $Lxx(x, \sigma)$ 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 $Lxy(x, \sigma)$, $Lyy(x, \sigma)$ e $Lxy(x, \sigma)$.

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 previamente filtrada, 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* 9×9 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, tendo em conta 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, para cada nova oitava, o tamanho do filtro é dobrado. 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), de forma a localizar pontos de interesse na imagem e sobre escalas, uma supressão não máxima numa vizinhança $3 \times 3 \times 3$ é aplicada. Os máximos do determinante da matriz Hessiana são então interpolados em escala e espaço da imagem.

A fim de ser invariante a rotação, Bay, Tuytelaars e Gool (2006) usam 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 essas respostas são calculadas e ponderadas com um Gaussi-

ano ($\sigma = 2,5s$) 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 de características 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 de 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 ponderados com uma Gaussiana para oferecer maior robustez para deformações, ruído e translação (BAY; TUYTELAARS; GOOL, 2006).

2.2.1.3 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; JOLLIFFE, 2014) e LDA (LI; YUAN, 2005). LSH (*Local Sensitive Hashing*) (KULIS; GRAUMAN, 2009) também é utilizado para converter esses descrito-

res 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 (ALAHI; 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) criam um vetor com as respostas dos testes, que são calculados após a suavização da região da imagem. Mais especificamente, define-se um t teste na região p de tamanho SxS como:

$$t(p; x, y) := \begin{cases} 1, & \text{se } p(x) < p(y) \\ 0, & \text{caso contrário} \end{cases} \tag{2.11}$$

Onde p(x) é a intensidade do pixel na após 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).

$$f_{nd}(p) = \sum_{1 \leqslant i \leqslant nd} 2^{i-1} t(p; x_i, y_i) \tag{2.12}$$

BRIEF utiliza uma região de imagem suavizada e seleciona um conjunto de pares (x, y). Em seguida, as comparações de intensidade de pixel são feitas nesses locais, resultando em 1 ou 0. Segundo Calonder et al. (2010), experimentos mostraram que apenas 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 fornecer-lhe.

2.2.1.4 ORB

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

Na fase de detecção de *keypoints*, Rublee et al. (2011) utilizam o algoritmo FAST para encontrar pontos de interesse, em seguida, aplica 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) utilizam 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 imputar uma orientação. Rosin (1999) define o momento da região como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.13}$$

E com esses momentos, pode-se encontrar o centroide:

$$C = \left(\frac{m_{10}}{m_{00}}, \quad \frac{m_{01}}{m_{00}}\right) \tag{2.14}$$

A orientação da região é dada por:

$$\theta = atan2(m_{01}, m_{10}) \tag{2.15}$$

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 circular 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) afirmam que qualquer conjunto de características de n testes binários no local (x_i, y_i), define a matriz 2 x n:

$$S = \begin{pmatrix} x_1, & \dots & , x_n \\ y_1, & \dots & , y_n \end{pmatrix} \tag{2.16}$$

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

$$S_\theta = R_\theta S \tag{2.17}$$

Então, a versão rotacionada de BRIEF é dada por:

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S\theta \tag{2.18}$$

Desta forma, Rublee et al. (2011) discretizam o ângulo a 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 Sθ será usado para calcular o seu descritor.

Assim, ao contrário BRIEF, ORB é comparativamente invariante a escala e rotação enquanto ainda emprega a eficiente distância de Hamming no processo de combinação (RUBLEE et al., 2011).

2.2.1.5 BRISK

A detecção de *keypoints* de BRISK (*Binary Robust Invariant Scalable Keypoints*) é inspirada em AGAST (MAIR et al., 2010), que, por sua vez, é uma extensão para desempenho acelerado do detector FAST (LEUTENEGGER; CHLI; SIEGWART, 2011).

As camadas da pirâmide escala-espaço consistem, tipicamente de 4 oitavas e 4 intra-oitavas. As oitavas são formados por progressivamente 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. Assim, cada intra-oitava está localizada entre camadas de oitavas (LEUTENEGGER; CHLI; SIEGWART, 2011).

Inicialmente, Leutenegger, Chli e Siegwart (2011) aplicam o detector FAST 9-16 em cada oitava e intra-oitava separadamente usando o mesmo limiar para identificar potenciais regiões de interesse, conforme apresentado na Figura 9. Em seguida, os pontos pertencentes a estas regiões são submetidos a uma supressão não-máxima em escala-espaço: 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.

Utilizando pontos obtidos por meio da supressão não-máxima, uma função quadrática 2D é determinada para ajustar a região 3×3 em torno dos pixeis e sub-pixeis máximos. O mesmo é feito para a camada acima e abaixo. Estes valores máximos são então interpoladas utilizando uma função quadrática 1D através do espaço escala e

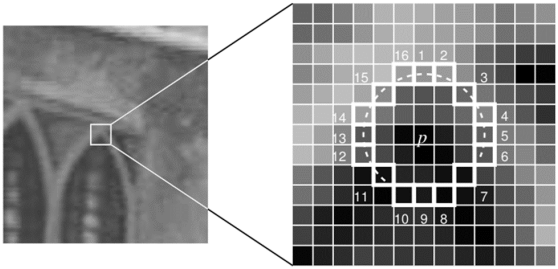


Figura 9 – O detector FAST de 16 pontos é usado, o que requer 9 pixeis consecutivos que são suficientemente mais brilhantes ou mais escuros do que o pixel central (GLEASON, 2011).

o máximo local é escolhido como a escala em que a característica foi encontrada (GLEASON, 2011).

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

Segundo Leutenegger, Chli e Siegwart (2011), o descritor BRISK faz uso de um padrão, ilustrado na Figura 10, que define locais igualmente espaçados em círculos concêntricos com o ponto de interesse. A fim de evitar os efeitos de *aliasing* no momento da amostragem, a intensidade da imagem de um ponto no padrão, BRISK aplica suavização Gaussiana com desvio padrão proporcional à distância entre os pontos sobre o respectivo círculo.

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

$$g(p_i, p_j) = (p_j - p_i)\frac{I(p_j, \sigma_j) - I(p_{i,\sigma_{ij}})}{\|p_j - p_i\|^2} \tag{2.19}$$

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 distancia ($\mathcal{L}$) de acordo com a escala do ponto de interesse. A distância limiar é definida proporcionalmente a escala do *keypoint*. Iterando através os pares de pontos em $\mathcal{L}$, Leutenegger, Chli e Siegwart (2011)

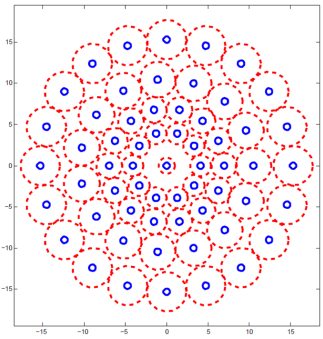


Figura 10 – O padrão de amostragem BRISK com N = 60 pontos: os pequenos círculos azuis indicam os locais de amostragem; os maiores, círculos vermelhos tracejados, são desenhados num raio σ correspondente ao desvio padrão do núcleo Gaussiano utilizado para suavizar os valores de intensidade nos pontos de amostragem. (LEUTENEGGER; CHLI; SIEGWART, 2011).

estimam a direção do ponto de interesse k como:

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{p_i, p_j \in \mathcal{L}} g(p_i, p_j) \tag{2.20}$$

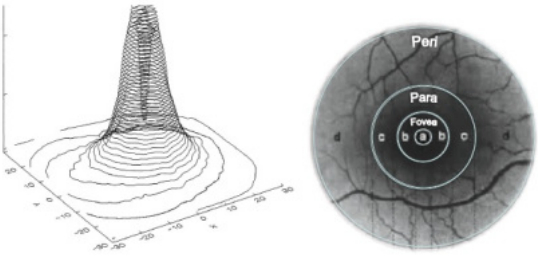
Para a formação do descritor com rotação e escala normalizados, Leutenegger, Chli e Siegwart (2011) aplicam o padrão de amostragem rotacionado por $\alpha = arctan2(g_y, g_x)$ em torno do ponto de interesse k. O descritor dk é um vetor de bits montado através da realização de todas as comparações intensidade nos pares de ponto (p_j^α, p_i^α) de S.

$$b = \begin{cases} 1, & \text{se } I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, & \text{caso contrário} \end{cases} \tag{2.21}$$

2.2.1.6 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 precisamente na retina. Uma cascata de cadeias binárias é calculada comparando intensidades de imagem ao longo de um padrão de amostragem similar ao da retina.

Muitas redes de amostragem são possíveis para comparar pares de intensidades de pixeis. Alahi, Ortiz e Vandergheynst (2012) utilizam uma malha de amostragem inspirada na retina humana que, assim como BRISK, é circular com a diferença de ter maior densidade de pontos perto do centro. A densidade de pontos cai exponencialmente como pode ser visto na Figura 11. Cada ponto de amostragem deve ser suavizado para tornar-se menos sensível ao ruído.



(a) Density of ganglion cells over the retina (b) Retina areas

Figura 11 – (a)Densidade de células ganglionares sobre a retina (b) Áreas da retina (ALAHI; ORTIZ; VANDERGHEYNST, 2012)

Para coincidir com o modelo de retina, Alahi, Ortiz e Vandergheynst (2012) utilizam núcleos de tamanhos diferentes para cada pontos amostral, assim como BRISK, porém com uma mudança exponencial do tamanho dos campos receptivos sobrepostos. A Figura 12 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 (ALAHI; ORTIZ; VANDERGHEYNST, 2012). Em outras palavras, o descritor é uma cadeia binária formada por uma sequência de Diferença entre Gaussianas de 1 bit:

$$F = \sum_{0 \leq \alpha < N} 2^{\alpha} T(P_{\alpha}), \tag{2.22}$$

Onde P_{α} é um par de campos receptivos, N é o tamanho desejado do descritor, e

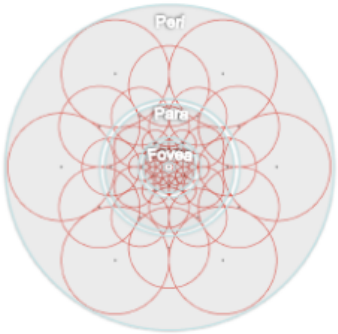


Figura 12 – Ilustração do padrão de amostragem FREAK em que cada círculo representa um campo receptivo onde a imagem é suavizada com o respectivo núcleo Gaussiano (ALAHI; ORTIZ; VANDERGHEYNST, 2012)

$$T(P_\alpha) = \begin{cases} 1, & \text{se } I(p_\alpha^{r_1}) - I(p_\alpha^{r_2}) > 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2.23}$$

Com $I(p_\alpha^{r_1})$ é a intensidade suavizada do primeiro campo receptivo do par p_α.

Com algumas dezenas de campos receptivos, milhares de pares são possíveis o que gera um grande descritor. 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 descritor 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 ponto chave, Alahi, Ortiz e Vandergheynst (2012) somam os gradientes locais dos pares selecionados similarmente a BRISK, porém selecionam pares com campos receptivos simétricos em relação ao centro. FREAK seleciona 45 pares em oposi-

ção às centenas de pares de BRISK. Além disso, o padrão de retina tem campos receptivos maiores na área perifoveal que BRISK permitindo um maior erro na estimativa de orientação.

2.2.2 RANSAC

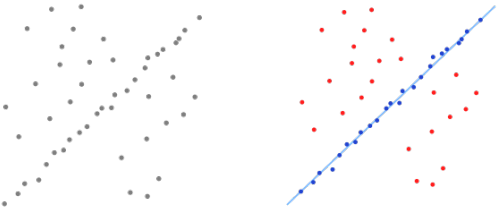
O algoritmo *RANdom SAmple Consensus* (RANSAC) proposto por Fischler e Bolles (1981) propõe uma abordagem de estimação de parâmetros gerais projetada para lidar com uma grande proporção de *outliers* nos dados de entrada. Ao contrário de muitas das técnicas de estimação robustas comuns que foram adotadas pela comunidade de visão computacional a partir da literatura estatística, RANSAC foi desenvolvido a partir da comunidade de visão computacional, sendo amplamente usado 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 "verdadeiro" instanciado pelo conjunto de parâmetros "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, 1981).

A Figura 13 apresenta uma aplicação simples do algoritmo RANSAC em um conjunto bidimensional de dados.

O algoritmo 1 abaixo sumariza o funcionamento do RANSAC.



(a) Representação visual do conjunto de dados

(b) *Inliers* e *outliers* RANSAC

Figura 13 – Aplicação simples do algoritmo RANSAC em um conjunto bidimensional de dados. Adaptado de (ISACK; BOYKOV, 2012)

Algoritmo 1 RANSAC

1: Seleciona aleatoriamente o número mínimo de pontos necessário para determinar os parâmetros do modelo.
2: Um modelo é ajustado ao subconjunto selecionado, ou seja, aos *inliers* hipotéticos;
3: Determina quantos pontos do conjunto de todos os pontos se encaixam com uma tolerância predefinida ϵ.
4: Se a fração de um número de *inliers* sobre o número total de pontos no conjunto exceder um limiar predefinido τ, re-estima os parâmetros do modelo usando todos os *inliers* identificados e termina.
5: Caso contrário, repete os passos 1 a 4 (máximo de N vezes).

2.2.3 Registro de imagens

Desde a introdução de sensores de profundidade de baixo custo, como o Microsoft Kinect, novos progressos têm sidos feitos no domínio robótico para localização simultânea e mapeamento (SLAM) (AULINAS et al., 2008)(ENDRES et al., 2012). O mapa reconstruído é representado por um conjunto de nuvens de pontos que são alinhadas por meio de registro e pode ser usado para evitar obstáculos, exploração de mapas, controle de veículos autônomos, etc. (SPRICKERHOF et al., 2009) (HUANG et al., 2011). Além disso, as informações de profundidade são muitas vezes combinadas com uma câmera tradicional RGB (NEWCOMBE et al., 2011) (RUHNKE et al., 2013), a fim de facilitar a resolução de problemas do mundo real, como a detecção de objetos em cenas desor-

denadas, rastreamento e reconhecimento de objetos (BELLEKENS et al., 2014).

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 modo 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 uma métrica de erro de distância. 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).

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

- Diferentes pontos de vista (análise multivista): 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;
- Diferentes tempos (análise multitemporal): As imagens da mesma cena são adquiridas em épocas diferentes, frequentemente em base regular, e possivelmente sob circunstâncias diferentes. O objetivo é encontrar e avaliar mudanças na cena que tenham ocorrido entre as aquisições consecutivas de imagens;
- Diferentes sensores (análise multimodal): As imagens da mesma cena são adquiridas por sensores diferentes. O objetivo é integrar as informações obtidas de diferentes fluxos de fonte para obter uma representação de 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 apresentado na Figura 14:

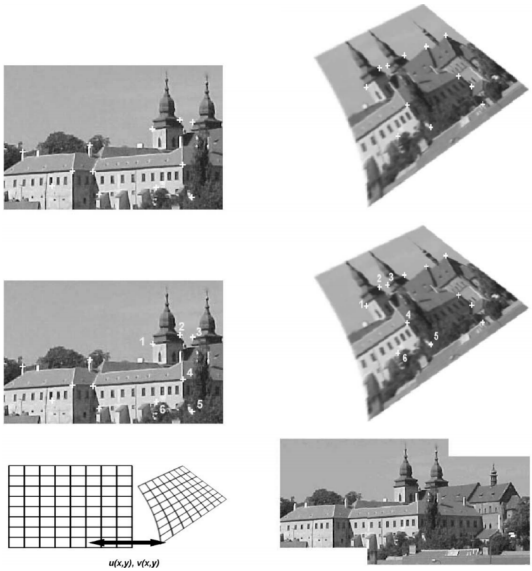


Figura 14 – Etapas do registro (cantos foram utilizados como características, neste caso) (ZITOVA; FLUSSER, 2003).

- Detecção de características: Objetos salientes e distintivos (regiões limítrofes, arestas, contornos, interseções de linhas, cantos, etc.) são manualmente ou preferencialmente automaticamente detectados. Para processamento posterior, essas características podem ser representadas por seus representantes pontuais (centros de gravidade, terminações de linha, pontos distintivos), chamados pontos de controle (CPs) na literatura.

- Correspondência entre características: Neste passo, é estabelecida a correspondência entre as características detectadas na imagem detectada e as detectadas na imagem de referência. Vários descritores de características e medidas de similaridade, juntamente

com relações espaciais entre as características são utilizados para esse fim.

- 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 calculados por meio da correspondência de característica estabelecida.

- 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.

Embora o processo de registro seja independente do alinhamento inicial das nuvens, em geral, os métodos baseados em características são mais lentos que os em ICP (BASDOGAN; OZTIRELI, 2008).

As próximas seções apresentarão definições de conceitos necessários para compreender e realizar o processo de registro.

2.2.3.1 Nuvens de pontos

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.

Point Cloud Library (PCL) é uma biblioteca de código aberto que 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).

Vários são os sensores capazes de gerar essas nuvens de pontos, por exemplo scanners 3D. 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 RGB-D, como Kinect, Ensenso Stereo 3D e DepthSense, que podem estar acopladas em uma variedade de dispositivos, incluindo robôs móveis.

O sensor Kinect é um dispositivo periférico (projetado inicialmente para jogos eletrônicos) que além de fornecer uma imagem RGB, também fornece um mapa de profundidade. Significado para cada pixel visto pelo sensor, o dispositivo também mede a distância ao sensor de acordo com suas limitações de distância (HÖGMAN, 2012). A Figura 15 apresenta o dispositivo.

Figura 15 – Sensor Microsoft Kinect. (a) O sensor Kinect para Xbox 360. (b) O Infravermelho (IR), câmera IR e câmera RGB dentro de um sensor Kinect (ZHANG, 2012).

A câmera RGB, que opera a 30 Hz, possui resolução em 640×480 pixeis com 8 bits por canal. Kinect também tem a opção de mudar a câmera para alta resolução, executando em 10 FPS em 1280×1024 pixeis. A própria câmera 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).

Cruz, Lucio e Velho (2012) afirmam que o sistema de detecção de profundidade 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 câmera IR opera a 30 Hz e possui resolução de 1200×960 pixeis. Estas imagens são subamostradas a 640×480 pixeis com 11-bits, o que fornece 2048 níveis de sensibilidade.

De posse dos valores de profundidade, pode-se 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 apresentado na Figura 16.

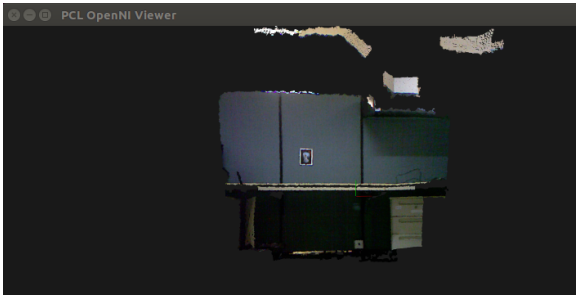


Figura 16 – Nuvem de pontos obtida pelo sensor Kinect com auxílio da biblioteca PCL

2.2.3.2 PFH

Em sua representação nativa, pontos como definidos no conceito de sistemas de mapeamento 3D são representados usando suas coordenadas cartesianas x, y, z em relação a uma dada origem. Assumindo que a origem do sistema de coordenadas não mude ao longo do tempo, pode haver dois pontos p1 e p2, adquiridos em tempos t1 e t2, com as mesmas coordenadas. Comparar esses pontos, no entanto, não é um problema trivial, porque mesmo que eles sejam iguais em relação a alguma medida de distância (euclidiana, por exemplo), eles poderiam ser amostrados em superfícies completamente diferentes e, portanto, representarem informações totalmente diferentes. Isso ocorre porque não há garantias de que o mundo não tenha mudado entre t1 e t2. Alguns dispositivos de aquisição podem fornecer informações adicionais para um ponto amostrado, como um valor de intensidade ou de remissão de superfície, ou até mesmo uma cor, porém isso não resolve o problema completamente e a comparação permanece ambígua (RUSU, 2010).

Aplicações que precisam comparar pontos por várias razões exigem melhores características e métricas para poder distinguir entre superfícies geométricas. O conceito de um ponto 3D como uma entidade singular com coordenadas cartesianas, portanto, desaparece, e um novo conceito toma o seu lugar, o de descritor local. A literatura é abundante em diferentes nomenclaturas descrevendo a mesma conceituação, como descritores de forma ou características geométricas, mas para o

restante deste documento eles serão referidos como representações de característica de ponto (RUSU, 2010).

De acordo com Rusu et al. (2008), curvas superficiais estimadas e normais para um ponto são duas das características de ponto mais amplamente utilizadas para representar nuvens de pontos, e desempenham um papel importante em aplicações como o registro e segmentação, por exemplo. Ambas são consideradas características locais, pois caracterizam um ponto usando as informações fornecidas pelos k vizinhos mais próximos do ponto. No entanto, seus valores são altamente sensíveis ao ruído do sensor e à seleção dos k vizinhos.

O descritor *Point Feature Histograms* (PFH) define características locais invariantes que representam as propriedades do modelo de superfície subjacente em um ponto p (RUSU et al., 2008).

Para obter eficientemente tais características, Rusu et al. (2008) propõem a computação e o uso de histogramas de valores que codificam as propriedades geométricas em uma vizinhança, fornecendo uma escala global e apresentando uma característica invariante de múltiplos valores. O histograma generaliza a curvatura superficial média em um ponto p e baseia-se na presença de coordenadas 3D e normais de superfície estimadas (RUSU et al., 2008) (RUSU; BLODOW; BEETZ, 2009).

Segundo Rusu et al. (2008), para cada ponto p, todos seus vizinhos fechados em uma esfera com um determinado raio r são selecionados, conforme apresentado na Figura 17. Se a superfície normal no ponto p está ausente, ela é substituída pela normal do melhor plano de ajuste para a superfície da vizinhança - normalmente realizado usando PCA:

$$n_{pi} = V_0, \lambda 0 \leq \lambda 1 \leq \lambda 2 \tag{2.24}$$

Onde V_0 é o autovetor correspondente ao menor autovalor $\lambda 0$.

Uma vez que todas as normais são obtidas, Rusu et al. (2008) utilizam as informações do ponto de vista existente para reorientá-las consistentemente. Ou seja, se:

$$\frac{(v - p_i), n_{pi}}{\|v - p_i\|} < 0 \tag{2.25}$$

Onde n_{pi} é o normal do ponto p_i e v é o ponto de vista, então $n_{pi} = -n_{pi}$;

Para cada par de pontos p_i e p_j $(i \neq j, j < i)$ na k-vizinhança de p, e suas normais estimadas n_i e n_j, Rusu et al. (2008) selecionam um ponto fonte p_s e um ponto alvo p_t - sendo a fonte a que tem o menor

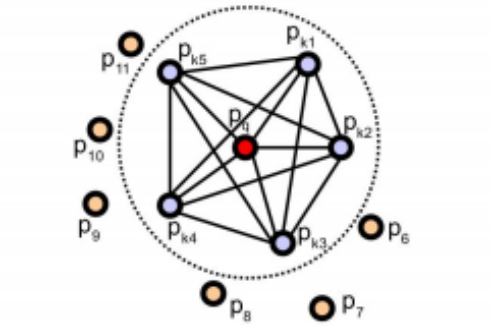


Figura 17 – O diagrama de região de influência para um PFH. O ponto (vermelho) e seus k-vizinhos (azul) estão totalmente interconectados em uma malha (RUSU; BLODOW; BEETZ, 2009).

ângulo entre a normal associada e a linha que liga os pontos:

$$\begin{aligned} \text{se } (n_i, p_j - p_i) \leq (n_j, p_i - p_j) \\ \text{então } p_s = p_i, p_t = p_j \\ \text{caso contrário } p_s = p_j, p_t = p_i \end{aligned} \tag{2.26}$$

E então define-se o quadro de Darboux, conforme apresentado na Figura 18, com a origem no ponto fonte como:

$$u = n_s, v = (p_t - p_s) \times u, w = u \times v \tag{2.27}$$

As quatro características são categorizadas usando um histograma de 16 compartimentos, onde cada compartimento no índice idx contém a porcentagem dos pontos de origem na vizinhança que têm suas características no intervalo definido por idx (RUSU et al., 2008) (RUSU, 2010):

$$\left.\begin{array}{c} f_1 = (v, n_t) \\ f_2 = \|p_t - p_s\| \\ f_3 = (u, p_t - p_s)/f_2 \\ f_4 = atan((w, n_t), (u, n_t)) \end{array}\right\} idx \sum_{i=1}^{i \leq 4} step(s_i, f_i).2^{i-1} \tag{2.28}$$

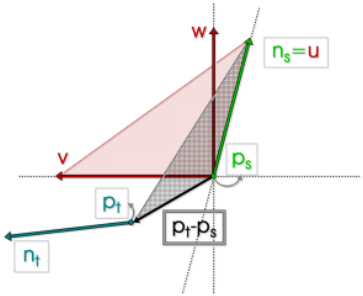


Figura 18 – O quadro de Darboux calculado (vetores u, v e w) colocado no ponto de origem (RUSU et al., 2008).

Onde *steps*(s, f) é definido como 0 se f<s e 1 caso contrário.

Isto significa que ao definir s_i como centro do intervalo de definição de f_i (ou seja, 0 para as características f1, f3, f4 e r para f2), o algoritmo classifica cada característica de um par p_i, p_j na vizinhança de p em duas categorias, e salva a porcentagem de pares que têm a mesma categoria para as quatro características (RUSU et al., 2008) (RUSU, 2010).

Rusu et al. (2008) afirmam que as quatro características são uma medida dos ângulos entre as normais dos pontos e o vetor de distância entre eles. Como f1 e f3 são produtos de ponto entre vetores normalizados, eles são o cosseno dos ângulos entre os vetores 3D, assim seu valor está entre ± 1. Similarmente, f4 é o arco tangente do ângulo que n_t forma com w se projetado no plano definido por u = n_t e w, então seu valor está entre $\pm \pi/2$.

2.2.3.3 FPFH

O objetivo de *Fast Point Feature Histograms* (FPFH) é reduzir a complexidade computacional do algoritmo PFH mantendo a maior parte de seu poder discriminativo (RUSU; BLODOW; BEETZ, 2009)(RUSU, 2010).

Para simplificar a computação de características do histograma, Rusu, Blodow e Beetz (2009) procedem da seguinte forma:

- Numa primeira etapa, para cada ponto de consulta p são calculadas apenas as relações (ver Equação 2.29) entre si e os seus vizinhos - chamado de *Simplified Point Feature Histogram* (SPFH);

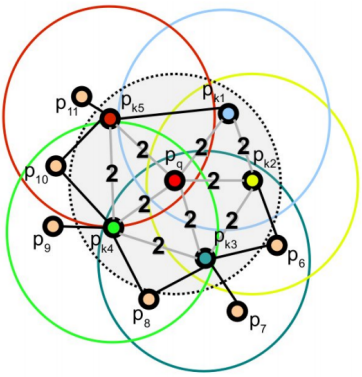


Figura 19 – O diagrama de região de influência para um FPFH (RUSU; BLODOW; BEETZ, 2009).

- Em uma segunda etapa, para cada ponto, seus k vizinhos são redeterminados e os valores de SPFH dos vizinhos são usados para ponderar o histograma final de p (chamado FPFH):

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}SPFH(p_k) \tag{2.29}$$

Onde o peso ω_k representa a distância entre o ponto de consulta p e um ponto vizinho p_k num dado espaço métrico.

A Figura 19 apresenta o diagrama de região de influência para um FPFH. Cada ponto de consulta (vermelho) está conectado apenas aos seus vizinhos k diretos (fechados pelo círculo cinza). Cada vizinho direto é conectado a seus próprios vizinhos e os histogramas resultantes são ponderados juntamente com o histograma do ponto de consulta para formar o FPFH. As conexões marcadas com 2 irão contribuir para o FPFH duas vezes. Para um dado ponto de consulta p_q, primeiro são estimados seus valores SPFH criando pares entre si e seus vizinhos. Isto é repetido para todos os pontos no conjunto de dados, e então os valores SPFH de p_k são reponderados usando os valores SPFH de seus vizinhos, criando assim o FPFH para p_q. As diferenças entre PFH e FPFH são (RUSU; BLODOW; BEETZ, 2009) (RUSU, 2010):

- FPFH não interconecta completamente todos os vizinhos de p_q, como pode ser visto nas Figuras 19 e 17, e não computa alguns pares de valores que podem contribuir para capturar a geometria

em torno de p_q;

- PFH modela uma superfície precisamente determinada em torno de p_q, enquanto que o FPFH inclui pares de pontos adicionais fora da esfera de raio r (embora, no máximo, 2r de distância);
- por causa do esquema de reponderação, FPFH combina os valores de SPFH e recapta alguns dos valores dos pares de pontos vizinhos.

2.2.3.4 SAC-IA

O problema de registro torna-se facilmente resolvível se as correspondências ponto a ponto são perfeitamente conhecidas nos conjuntos de dados de entrada. Isto significa que uma lista selecionada de pontos $p_i \in P$ tem que "coincidir" de um ponto de vista de representação de característica com outra lista de pontos $q_j \in Q$, onde P e Q representam dois conjuntos de dados de vista parcial (RUSU, 2010).

O método *Sample Consensus Initial Aligment* (SAC-IA) foi apresentado por Rusu, Blodow e Beetz (2009) como um método de alinhamento inicial entre duas nuvens P e Q para registro que tenta manter as mesmas relações geométricas das correspondências sem ter que tentar todas as combinações de um conjunto limitado de correspondências. Em vez disso, SAC-IA amostra um grande número de candidatos a correspondência e pode os classificar rapidamente, empregando o seguinte esquema (BONNAL, 2010) (RUSU; BLODOW; BEETZ, 2009):

- Seleciona pontos de amostra de P enquanto se certifica de que suas distâncias pareamento são maiores do que uma distância mínima;
- Para cada um dos pontos de amostragem, encontra uma lista de pontos em Q cujos histogramas sejam semelhantes ao histograma dos pontos de amostra. A partir destes, seleciona um aleatoriamente que será considerado a correspondência da amostra de pontos;
- Calcula a transformação rígida definida pelos pontos de amostragem e suas correspondências e uma métrica de erro para a nuvem de pontos que serve como métrica da qualidade da transformação.

A seleção aleatória das correspondências no segundo passo pode parecer subótima. No entanto, Rusu (2010) argumenta que é uma

decisão que pode ser feita muito rapidamente, enquanto uma estratégia mais complicada que tenta maximizar a semelhança do histograma e semelhanças de distância pode muito provavelmente exigir múltiplas passagens sobre as listas de candidatos a correspondência.

O passo final proposto por Rusu, Blodow e Beetz (2009) é calcular a transformação rígida definida pelos pontos de amostragem e suas correspondências e calcular uma métrica de erro para a nuvem de pontos que calcula a qualidade da transformação.

Rusu, Blodow e Beetz (2009) propõem usa como métrica de erro para a esta etapa é uma medida de penalidade Huber L_h:

$$L_h(e_i) = \begin{cases} \frac{1}{2}e_i^2 & \|e_i\| \leq t_e \\ \frac{1}{2}t_e(2\|e_i\| - t_e) & \|e_i\| > t_e \end{cases} \tag{2.30}$$

2.2.3.5 ICP

No *Iterative Closest Point* (ICP), uma forma de dados P é deslocada (registrada, posicionada) para entrar em melhor alinhamento com uma forma modelo X (BESL; MCKAY, 1992). Assim, o algoritmo ICP visa encontrar a transformação entre duas superfícies (duas nuvens de pontos, por exemplo) minimizando os erros quadráticos entre as entidades correspondentes.

O ICP é frequentemente usado para reconstruir superfícies 2D ou 3D a partir diferentes fontes criando um modelo, que pode ser usado para localizar robôs e alcançar o planejamento ótimo do caminho (especialmente quando a odometria de roda não é confiável devido ao terreno escorregadio), co-registrar modelos ósseos, dentre outros (CHEN; MEDIONI, 1992).

De forma geral, o funcionamento do algoritmo ICP no problema de alinhamento de nuvens de pontos é descrito a seguir (BESL; MCKAY, 1992):

- Para cada ponto da nuvem de pontos de origem, o ICP localiza o ponto mais próximo na nuvem de pontos de referência. A distância d entre um ponto de dados $\vec{p}$ e a forma modelo X é denotada por:

$$d(\vec{p}, X) = \min_{\vec{x} \in X} \|\vec{x} - \vec{p}\| \tag{2.31}$$

  O ponto mais próximo em X que possui uma menor distância é denotado $\vec{y}$ tal que $d(\vec{p}, \vec{y}) = d(\vec{p}, X)$ onde $\vec{y} \in X$. Assim, seja Y o conjunto resultante de pontos mais próximos, e seja C o operador

do ponto mais próximo:

$$Y = C(P, X) \tag{2.32}$$

- Dado o conjunto de pontos Y resultante, o próximo passo é estimar a combinação de rotação e translação usando mínimos quadrados, de forma que melhor alinhe cada ponto de origem a sua correspondência encontrada na etapa anterior (minimize as distancias entre p e x).

$$(\vec{q}, d) = Q(P, Y) \tag{2.33}$$

- O ICP aplica a transformação obtida na etapa anterior:

$$P_{k+1} = \vec{q}_k(P_0) \tag{2.34}$$

- Por fim, o algoritmo deve iterar (reassociar os pontos, e assim por diante). O término ocorre quando a iteração obtiver um erro abaixo de um limite predefinido

Desta forma, o algoritmo padrão ICP busca resolver um problema de minimização de erro L_2 (YANG; LI; JIA, 2013).

Yang, Li e Jia (2013) afirmam que sejam dois conjuntos de pontos 3D $X = x_i$, $i = 1, ..., M$ e $Y = y_j$, $j = 1, ..., N$, onde x_i e $y_j \in \mathbb{R}^3$ são coordenadas dos pontos, sendo uma delas a fonte e a outra modelo, o objetivo do algoritmo é estimar um movimento rígido composto por rotação $R \in SO(3)$ e translação $t \in \mathbb{R}^3$ que minimize o seguinte erro L_2:

$$E(R, t) = \sum_{i=1}^{M} e_i(R, t)^2 = \sum_{i=1}^{M} \|(Rx_i + t - y_{j*})\|^2 \tag{2.35}$$

onde $e_i(R, t)$ é o erro residual por ponto para x_i. O ponto $y_{j*} \in Y$ é denotado como a correspondência ótima de x_i, que, no contexto do ICP, é o ponto mais próximo da transformação x_i em Y, por exemplo:

$$j* = \underset{j \in \{1,...,N\}}{\operatorname{argmin}} \|(Rx_i + t - y_j)\| \tag{2.36}$$

Dada uma transformada inicial R e t, o algoritmo ICP resolve a minimização acima alternando entre estimar a transformação com base na Equação 2.35 e encontrar as correspondências de ponto mais

próximo pela Equação 2.36. Devido a essa natureza iterativa, o ICP só pode garantir a convergência para um mínimo local (YANG; LI; JIA, 2013) (BESL; MCKAY, 1992).

Existem muitas variantes ICP, que enfocam as diversas fases do algoritmo, buscando maior correspondência entre pontos, velocidade ou até mesmo ambos (RUSINKIEWICZ; LEVOY, 2001).

2.3 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 resolvidos com o auxílio de técnicas de visão computacional.

Além disso, foram apresentados uma série de descritores de características locais, úteis no processo de encontrar marcações naturais que podem ser utilizadas nas fases de mapeamento e localização de robôs móveis. Por fim, foi apresentado o conceito de registro de nuvens de pontos que visa construir um modelo global do ambiente percorrido por um robô móvel, bem como suas principais abordagens e partes integrantes.

3 TRABALHOS RELACIONADOS

O constante desenvolvimento de técnicas de visão computacional contribui para encontrar soluções para as mais diversas áreas, dentre elas a robótica móvel. 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 evolução da área.

A principal vantagem de sistemas baseados em visão computacional está na grande quantidade de informação que estes disponibilizam. Porém o tempo de processamento computacional necessário para a compreensão e conversão desta informação em dados úteis é, normalmente, bastante elevado. 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 E MAPEAMENTO

Navegação e posicionamento de robôs móveis com sistemas de visão são geralmente realizados com o auxílio de referências naturais ou artificiais (ALEKSANDROVICH et al., 2013). Se, Lowe e Little (2002) propõem utilizar o descritor SIFT para encontrar pontos de referência naturais com seu robô Triclops stereo. Utilizando a estimativa da odometria, o algoritmo localiza esses pontos em quadros consecutivos enquanto o robô se move, e estima o movimento da câmera (e do robô) utilizando mínimos quadrados nas marcações combinadas.

A abordagem de Saurer, Fraundorfer e Pollefeys (2010) se baseia no reconhecimento de lugares utilizando recursos visuais obtidos pelo descritor GIST (OLIVA; TORRALBA, 2001) para encontrar semelhanças entre imagens. Verificação geométrica através de pontos de fuga é utilizada para garantir que a imagem de banco do dados correspondente foi feita a partir do mesmo ponto de vista que a imagem obtida a partir do robô. O reconhecimento do local ocorre comparando a imagem recém capturada com as do banco de dados previamente armazenadas. Também utilizando o descritor GIST, Chang, Siagian e Itti (2010) construíram um sistema de localização que utiliza GIST e regiões salientes para localizar precisamente o robô, enquanto o sistema de navegação utiliza as regiões salientes para realizar o controle visual guiar o robô para atingir uma meta posição e orientação. GIST é usado para classi-

ficar uma imagem de entrada para um segmento, geralmente uma parte de um corredor, caminho ou estrada interrompida por um cruzamento ou uma barreira física em ambas as extremidades para uma delineação natural. A correspondência é feita através de pontos de interesse SIFT extraídos dentro de cada região saliente.

Chi et al. (2013) constroem um método à base de visão para mapear e localizar o robô móvel em um ambiente. Inicialmente, para construir o mapa, o robô se move aleatoriamente no ambiente alvo e captura imagens. SURF é utilizado para extrair as características de cada imagem. O mapa é construído através das características SURF e as suas posições. Para a fase de localização, uma nova imagem é obtida em cada local desconhecido quando o robô se move no ambiente. SURF e FLANN (*fast library for approximate nearest neighbors*) são utilizados para comparar as novas características da imagem com o mapa para inferir a posição do robô.

Garcia-Fidalgo e Ortiz (2013) apresentam uma proposta para mapeamento e localização visual baseada em reconhecimento de locais. A medida similaridade entre imagens é realizada com base no número de correspondências e suas distâncias associadas com o descritor SURF. Por outro lado, para otimizar os tempos de execução, as correspondências entre a imagem atual e os locais visitados anteriormente são determinadas usando um índice baseado em um conjunto de árvores k-d aleatórias. Além disso, um filtro Bayes discreto é usado para prever os candidatos a fechamento de loop (*loop closure problem* (COUTSIAS et al., 2004)), levando em consideração as relações anteriores entre locais visitados.

Outra perspectiva é proposta por Naseer et al. (2014), que resolveram o problema localização visual utilizando sequências de imagens recolhidas ao longo de rotas. Para conseguir uma localização robusta, foi utilizado uma descrição da imagem baseada numa rede densa de descritores de HOG (histograma de gradientes orientados) (DALAL; TRIGGS, 2005). Através da construção de um grafo de dados que relaciona imagens de sequências recuperadas em diferentes épocas do ano, o problema de reconhecimento visual do lugar visitado é resolvido através da computação de fluxos de redes no grafo associado, gerando múltiplas rotas.

3.2 AVALIAÇÃO DE DETECTORES E DESCRITORES DE CARACTERÍSTICAS LOCAIS

Mikolajczyk et al. (2005) comparam o desempenho de descritores computados para regiões de interesse. A avaliação utiliza como critério de comparação a precisão com relação ao *recall* (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, 1991) (KE; SUKTHANKAR, 2004), PCA-SIFT, invariantes diferenciais (*differential invariants*) (KOENDERINK; DOORN, 1987), *spin images* (LAZEBNIK; SCHMID; PONCE, 2003), SIFT, filtros complexos (SCHAFFALITZKY; 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) comparam os *trade-offs* (relação de compromisso) entre precisão e velocidade de BRIEF, BRISK, ORB, LIOP (Zhenhua 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 arvores k-d randomizados 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) comparam 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 métricas de desempenho. Os autores concluem que a partir dos resultados de obtidos, BRISK é altamente recomendável, possuindo a maior quantidade de melhores correspondências. Por outro lado, FREAK oferece apenas um desempenho um pouco menor, mas com uma computação mais rápida.

3.3 REGISTRO

Mitra et al. (2004) formulam o problema de alinhar nuvens de pontos como uma minimização de distâncias quadráticas entre superfícies subjacentes. Dado um par de nuvem de pontos, o algoritmo proposto encontra a transformada mais adequada solucionando um sistema linear obtido através de aproximações quadráticas da função de diferenças quadráticas de uma nuvem de pontos.

Rabbani e Heuvel (2005) apresentam um método para o registro automático de nuvens de pontos fazendo uma pesquisa restrita para encontrar os objetos, que podem ser modelados usando por outros mais simples como planos, cilindros e esferas, e utiliza-os como alvos. Uma vez que as correspondências entre objetos são encontradas automaticamente, o algoritmo encontra um registro aproximado. O registro final é baseado em um ajuste integrado de mínimos quadrados, onde os parâmetros de forma e pose de todos os modelos, bem como os parâmetros de registro de todas as varreduras são ajustados em uma única etapa. Esta metodologia não é geral e não pode ser aplicada a todas as cenas, porém pode reduzir o trabalho manual no registro de nuvens de pontos.

Sharp, Lee e Wehe (2002) utilizam características invariantes euclidianas em uma generalização do registro iterativo de ponto mais próximo. As correspondências pontuais são escolhidas como o ponto mais próximo em relação a uma combinação linear ponderada de distâncias posicional e característica. Sharp, Lee e Wehe (2002) mostram que em condições ideais livres de ruído, as correspondências formadas usando esta função de distância estão corretas com mais frequência do que as correspondências formadas usando a distância de posição sozinha.

A variante do algoritmo *Iterative Closest Point*, proposta por Men, Gebre e Pochiraju (2011), combina dados de intervalo de pontos normalizados e valores de tonalidade (matiz) ponderada calculados a partir de dados RGB de uma nuvem de pontos 3D. Uma busca de vizinhança mais próxima baseada em árvore k-d é usada para pontos comuns associados em x, y, z, matiz no espaço 4D. A matriz rígida de translação e rotação desconhecida necessária para o registro é iterativamente resolvida usando o método de Decomposição de Valor Singular (SVD). A maior parte do tempo e custo de computação durante o processo ICP é gasto na tentativa de encontrar pares de pontos corretos. No algoritmo ICP, os pontos correspondentes são pesquisados de acordo com a regra de distância mais próxima. Isso pode causar uma correspondência incorreta durante um único loop de iteração o que faz com que seja necessária mais de uma iteração para emparelhar os pontos

vizinhos próximos. Com base na propriedade de matiz correta, o melhor modelo pode ser encontrado em uma iteração. Assim, dependendo das informações de matiz corretas, o algoritmo proposto pode convergir com menos iterações (MEN; GEBRE; POCHIRAJU, 2011)

A abordagem proposta por Sehgal, Cernea e Makaveeva (2010) permite que o descritor SIFT encontre características locais em nuvens de pontos 3D e utiliza as correspondências entre pontos conhecidos para permitir o registro. Para extrair as características SIFT das informações de profundidade das nuvens de ponto, essa informação é colocada em um escala entre 0-255. A distância euclidiana para cada indivíduo (x, y, z) dentro da nuvem de pontos é calculada a partir da origem e colocada em escala usando escalas de raiz quadrada. Após a realização desta operação, os dados são passados através de um conversor PNG, a fim de obter imagens para as quais o operador SIFT é aplicado. Após a obtenção dos descritores de característica SIFT a partir das imagens, as correspondências entre as coordenadas (x, y, z) nas nuvens de pontos são obtidas pesquisando os descritores SIFT correspondentes, usando o algoritmo RANSAC. Cada ponto correspondente nas nuvens de ponto de profundidade pode ser representado por um quatérnion com coordenadas c = 0 e x, y e z, dadas pelas respectivas coordenadas do ponto na nuvem de pontos 3D. Ao comparar dados de duas nuvens pontuais consecutivas, Sehgal, Cernea e Makaveeva (2010) conseguem recuperar o quatérnion da matriz de rotação, que pode ser obtida a partir do quatérnion de rotação. Tendo obtido a matriz de rotação, pode-se obter o quatérnion de translação e o fator de escala, e aplicá-los às nuvens para realizar o registro.

3.4 CONCLUSÃO

Em geral, o uso de descritores de características locais para encontrar pontos de referência naturais tem se demonstrado como uma técnica satisfatória, encontrando resultados desejáveis para as tarefas de localização e mapeamento. Diversos descritores têm sido testados e utilizados nestas tarefas, com destaque para o descritor SIFT e suas derivações (SURF, por exemplo) e para os descritores binários, que utilizam a eficiente distância de Hamming para encontrar correspondências.

O problema de registro tem sido extensivamente estudado por pesquisadores, ressaltando o bom desempenho do algoritmo ICP e suas variantes, que por outro lado devem procurar maneiras de reduzir o

problema de mínimo local.

4 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA DE LOCALIZAÇÃO, MAPEAMENTO E REGISTRO 3D

Este capítulo apresenta a construção dos sistemas de localização, mapeamento e registro 3D. Inicialmente, foi realizado um estudo sobre a aplicabilidade de alguns descritores de características locais para o problema de localização robótica na forma proposta por este trabalho. Os tópicos seguintes abordam os recursos necessários, bem como as respectivas características e estruturas de funcionamento do sistemas propostos.

4.1 SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO

O algoritmo proposto visa localizar um robô ActivMedia Pioneer 3 enquanto ele percorre sua trajetória através das imagens capturadas por sua câmera utilizando descritores de características locais. Para verificar a aplicabilidade dos descritores a serem usados para o problema de imagens sequenciais, inicialmente foram realizados uma série de testes que serão descritos ao longo deste capítulo.

Armazenar e analisar todas as imagens percebidas por um robô durante uma tarefa de localização e mapeamento visual é tipicamente intratável para cenários reais. Para reduzir o número de imagens a considerar, algumas delas, chamadas *keyframes*, são selecionadas. Esses quadros-chave representam locais visualmente distintos do ambiente. No mapa proposto, cada nó corresponde a um quadro-chave, e cada quadro-chave é representado por seus descritores de características locais.

O software de localização é divido em duas etapas: de treinamento e mapeamento, e de localização. Na primeira etapa, o robô deve navegar em sua área trabalho seguindo sua trajetória para coletar imagens que serão utilizadas para determinar *keyframes* - pontos-chave nos quais o robô deve se localizar - que formarão os nós de um mapa topológico. Ou seja, durante a trajetória o robô deve encontrar com a ajuda dos descritores algumas localizações chave, as quais espera-se que ele esteja ao menos próximo enquanto percorre sua trajetória, para poder se localizar. Esta é, portanto, a etapa de treinamento e mapeamento. A segunda etapa é a de localização propriamente dita, em que as imagens recém-adquiridas enquanto o robô percorre sua trajetória são compa-

radas com os *keyframes* para determinar a localização aproximada do robô em relação aos nós do mapa topológico.

4.1.1 Recursos

Os resultados foram computados no conjunto de dados TUM RGB-D benchmark (STURM et al., 2012), especificamente, no conjunto fr2/pioneer_slam, que consiste em uma sequência de imagens RGB e de profundidade juntamente com estimativas de pose reais do robô obtidas a partir de um sistema de captura de movimento que representam o robô se movendo através de um ambiente interior.

Conjuntos de dados públicos apoiam eficientemente a avaliação científica e comparação objetiva de algoritmos. Vários exemplos de conjuntos de dados bem-sucedidos na área de visão computacional, como os conjuntos de dados Freiburg, Intel, Rawseeds e Newcollege (SMITH et al., 2009) (BONARINI et al., 2006) (KRETZSCHMAR; STACHNISS, 2012) usados para localização e mapeamento robótico, demonstraram que conjuntos de dados comuns e métricas de avaliação claras podem ajudar significativamente a evolução do estado da arte (STURM et al., 2012).

Além de sua ampla utilização por pesquisadores em sistemas RGB-D SLAM, este conjunto de dados foi selecionado pois fornece o *groundtruth* (medida da pose em cada local associada a cada instante de tempo em sua trajetória) obtido através um sistema de captura de movimento externo com oito câmeras de rastreamento de alta velocidade (100 Hz). Estes dados são fundamentais para a avaliação dos sistemas a serem propostos e para a associação de cada imagem com uma respectiva pose naquele instante de tempo. Por fim, outro fator importante são as claras e objetivas métricas propostas pelos autores para avaliação de sistemas a serem desenvolvidos, como será apresentado na seção 4.1.4.

O movimento do Kinect não é estritamente restrito a um plano porque tremores ocasionais (como resultado de colisões e fios no chão) desviam a orientação do equipamento. Todos os dados foram gravados em resolução total 640×480 e taxa de quadros completa do sensor Microsoft Xbox Kinect (30 Hz) em um laptop Linux rodando Ubuntu 10.10 e ROS Diamondback. Para acessar as imagens de cor e profundidade, foi usado o pacote de câmera OpenNI no ROS que envolve internamente o driver OpenNI da PrimeSense. Além disso, o driver kinect aux foi usado para registrar os dados do acelerômetro do Kinect a 500 Hz (STURM et al., 2012).

O Kinect foi montado em um ActivMedia Pioneer 3, conforme Figura 20 e se movimenta através de um labirinto de mesas, recipientes e outras paredes.

Figura 20 – Robô Pioneer com sensor Kinect (STURM et al., 2012)

Para obter a pose da câmera do sensor Kinect, um sistema de captura de movimento externo da MotionAnalysis foi usado. A configuração consistiu em oito câmeras Raptor-E com uma resolução de câmera de 1280×1024 pixeis em até 300 Hz (Figura 21). O sistema de captura de movimento rastreia a posição 3D dos marcadores passivos por triangulação. Para melhorar o contraste destes marcadores, as câmeras de captura de movimento são equipadas com LEDs infravermelhos para iluminar a cena (STURM et al., 2012).

O *groundtruth* é fornecido através de um arquivo de texto contendo a trajetória real armazenada por meio dos dados de tempo, vetor de translação e quatérnions unitários (formato: tempo tx ty tz qx qy qz qw). Esses dados são obtidos através da calibração e sincronização do Kinect com o sistema de captura de movimento externo. O erro relativo em uma base quadro a quadro nos dados "verdadeiros" do solo é inferior a 1 mm e 0,5 graus medidos no centro óptico do Kinect. Além disso, o erro absoluto em toda a área de captura de movimento é inferior a 10 mm e 0,5 graus (STURM et al., 2012).

O software de mapeamento e localização do robô foi feito em C++. A biblioteca de processamento de imagens OpenCV (BRADSKI et al., 2000) na versão 3.0 foi usada para extrair todos os *keypoints*, descritores e para implementação do algoritmo RANSAC. O motivo pelo qual foi escolhida esta biblioteca é grande quantidade existente de algoritmos e interfaces de comunicação com dispositivos periféricos que

Figura 21 – Sistema de captura de movimento MotionAnalysis usado para rastrear a pose da câmera do Kinect (STURM et al., 2012)

ela proporciona, dentre eles, destaca-se a interface com ROS (*Robot Operating System*), que fornece bibliotecas e ferramentas para ajudar os desenvolvedores de software a criar aplicativos para robôs (QUIGLEY et al., 2009).

O comprimento da trajetória é de 40,380 metros, com dimensões $5,50m \times 5,94m \times 0,07m$. As velocidades médias translacional e angular do robô são de 0,261 m/s e 13,379 graus/s respectivamente.

4.1.2 Avaliação de descritores de características para localização em robótica móvel

Esta etapa consiste em realizar uma série de testes para verificar a aplicabilidade dos descritores de características locais selecionados no problema de imagens sequenciais, para, então, poderem ser usados na localização em robótica móvel.

São usados os descritores SIFT, SURF, BRIEF, ORB, BRISK e FREAK e cada detector é combinado com o seu respectivo descritor. Ou seja, o extrator SIFT usa detector SIFT, por exemplo, exceto para os descritores FREAK e BRIEF, que usam os *keypoints* SIFT. A escolha destes descritores foi baseada nas avaliações anteriores encontradas na literatura (seção 3.2).

Para comparar de forma justa os descritores binários, SIFT e SURF, foi adotado o método de força bruta, que seleciona o descritor de uma característica no primeiro conjunto e a combina com todas as outras características do segundo conjunto usando algum cálculo de distância, no caso das distâncias euclidianas para os descritores SIFT e SURF ou de Hamming para os binários, e a mais próxima é retornada, gerando, assim, a combinação dos descritores. A figura 22 mostra o resultado da operação de correspondência entre duas imagens sequenciais do percurso do robô utilizando o descritor SIFT.

Figura 22 – Correspondências obtidas pelo método força bruta com descritor SIFT

Para cada descritor de característica em uma imagem, este método busca uma única correspondência na outra imagem. As linhas não paralelas na Figura 22 representam falsas correspondências. É fácil notar que este processo gera um alto número de correspondências incorretas.

O processo de correspondência é, então, melhorado primeiramente ajustando uma distância de limiar (*threshold*) usando as distâncias euclidiana ou de Hamming para os descritores SIFT e SURF, e binários respectivamente. O valor desta distância limiar para cada descritor foi determinado de forma empírica, observando que uma alta distância gera um alto número de falsos positivos enquanto um limiar baixo gera falsos negativos. A Figura 23 apresenta o resultado da operação de filtragem de correspondências entre as imagens acima após o processo de correspondência por força bruta.

A operação anterior elimina grande parte das falsas correspondências, porém, não todas. A implementação do RANSAC do OpenCV foi utilizada para eliminar as correspondências incorretas remanescentes. A Figura 24 apresenta o resultado final após RANSAC.

Enquanto o robô se move, é fundamental que o descritor seja

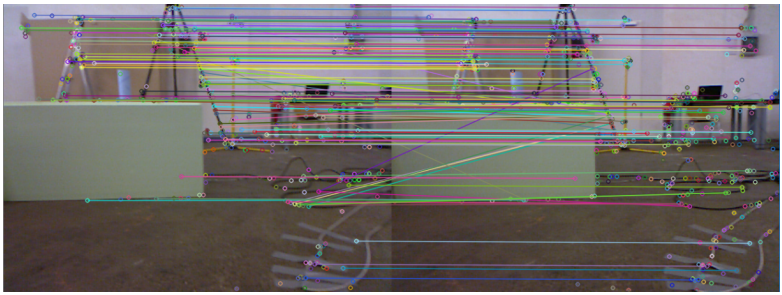


Figura 23 – Correspondências filtradas por distância limiar com descritor SIFT

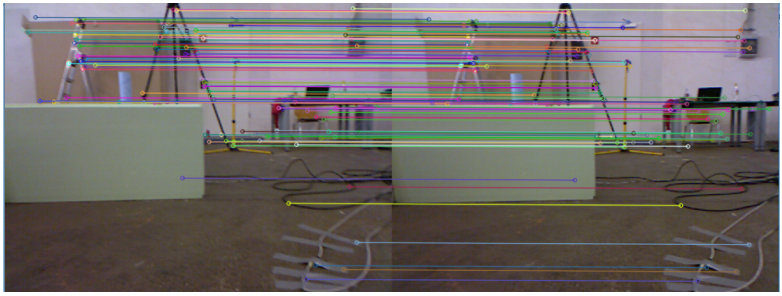


Figura 24 – Aplicação de RANSAC após etapas anteriores

capaz de distinguir a imagem recém obtida de todo conjunto de dados, principalmente em imagens muito similares para poder determinar a localização e os *keyframes* com maior precisão. Para demonstrar a robustez de cada descritor neste tipo de aplicação, a imagem recém obtida é comparada com sua vizinhança. A curva de resposta esperada deve ser semelhante a uma distribuição normal, com a imagem-alvo sendo a que tem mais correspondências e quanto mais longe, menos correspondências são esperadas se o robô não retorna a uma posição representada por algum dos quadros vizinhos.

Também é analisado se os sistemas propostos encontram qualquer ponto *outlier* através de todo o caminho que pode ser usado erroneamente para localização, especialmente no problema global, comparando uma imagem alvo com todo o conjunto de dados e retornando os quadros com mais correspondências.

As medições de distância podem geralmente ser definidas como medidas de similaridade e dissimilaridade; onde a primeira é definida de

acordo com o grau de similaridade entre suas instâncias, e a segunda, de acordo com as diferenças nos atributos das instâncias (WITTEN; FRANK; HALL, 2011). A distância euclidiana é mais comumente usada para medir a distância entre dois pontos ou partes (SARKER; ISLAM, 1999), e é usada para medir o grau de dissimilaridade que cada descritor encontra entre quadros próximos e é definida como a soma das raízes quadradas da diferença entre valores em suas respectivas dimensões, como demonstrado na Equação 4.1, onde $X_{a,j}$ é o valor da variável j para o individuo a e, analogamente, $X_{b,j}$ se refere ao valor da variável j para o individuo b.

$$d_{a,b} = \sqrt{\sum_{j=1}^{p}(X_{a,j} - X_{b,j})^2} \tag{4.1}$$

A localização do robô é, em geral, uma aplicação em que o tempo é um fator limitante. Encontrar a localização do robô uma vez que o robô tenha se movido uma distância considerável (vários metros, por exemplo) pode não ser de grande ajuda, visto que o contexto em que robô se encontra pode ser outro. O tempo de correspondência médio entre o quadro objetivo e o restante do seu ambiente também é testado para avaliar o desempenho de velocidade de correspondência entre quadros destes descritores. Esta etapa consiste em comparar o quadro teste com o restante dos quadros e extrair o tempo médio de comparação entre dois quadros.

O sistema de amostragem de quadros para a realização de testes foi baseado em seleção probabilística aleatória simples. Assim, utilizando-se de um nível de confiança de 95% com margem de erro 2% (MONTGOMERY; RUNGER; CALADO, 2000), foram selecionados aleatoriamente 16 quadros pertencentes à trajetória do robô (conjunto de dados fr2/pioneer_slam). Para a realização da etapa local, estes quadros foram comparados com seus dez próximos e dez anteriores vizinhos através de correspondência entre seus descritores, conforme métodos descritos acima.

4.1.3 Estrutura do algoritmo de localização e mapeamento

Após a realização da etapa de avaliação de descritores, verificou-se que os descritores SIFT, SURF, BRIEF, ORB e BRISK possuem as características adequadas para realizar a tarefa de localização e mapeamento na forma proposta por este trabalho.

O software de localização e mapeamento possui as etapas de treinamento e mapeamento, e de localização. A primeira etapa consiste em percorrer o conjunto de imagens e selecionar os *keyframes* com suas respectivas poses (vetor de translação e quatérnions). Os *keyframes* definem os nós de um mapa topológico. A utilização de mapas topológicos no projeto se justifica como representação adequada, pois utilizar todo o conjunto de imagens para tentar localizar o robô não seria eficiente em cenários reais devido ao grande número de imagens capturas por um robô durante sua trajetória. Além disso, trata-se de uma representação de mais simples e compacta (SANCHES, 2009).

Na fase de localização, a medida que a informação de vídeo é recebida enquanto o robô se move, o último quadro obtido é comparado com todo conjunto de *keyframes* definidos, determinando assim a localização global aproximada do robô no mapa proposto.

O fluxograma apresentado pela Figura 25 sumariza o funcionamento geral do sistema, conforme explicado acima.

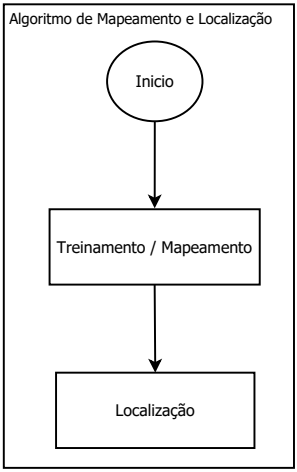


Figura 25 – Fluxograma geral de funcionamento

4.1.3.1 Etapa de treinamento e criação do mapa topológico

Esta etapa consiste em determinar quais quadros são quadros-chave ou *keyframes*. Para isso, define-se o primeiro quadro como *keyframe* e compara-o com os próximos através do uso de descritores de características locais.

O sistema de correspondência entre imagens dos descritores de características locais funciona de forma idêntica à etapa de avaliação, descrita em 4.1.2, com as etapas de correspondência, filtragem de boas correspondências por distâncias limiares e RANSAC.

O processo de determinação de detecção de *keyframes* ocorre pelo número de correspondências entre um quadro com seus vizinhos, que é a medida de similaridade entre quadros adotada por este trabalho. Quando o número de correspondências entre o *keyframe* e um dos próximos quadros chega a um limiar definido experimentalmente do número total de *keypoints* do *keyframe*, este quadro passa a ser um candidato a *keyframe*. Para ser um *keyframe*, é necessário que esse quadro não seja similar aos outros *keyframes*, visto que o robô pode passar por mais de uma vez em determinada localização. Esse quadro, então, passa a ser um *keyframe* após ser comparado com o conjunto de *keyframes* já definidos e verificar-se que eles não são semelhantes, com a mesma medida de similaridade proposta acima. O processo se repete até o fim da sequência de imagens, que identifica o fim da trajetória do robô, comparando sempre o novo *keyframe* com os próximos quadros e *keyframes* anteriores.

Cada *keyframe* representa um nó de um mapa topológico com as respectivas informações de pose (o vetor de translação e quatérnions), conforme representado pela Figura 26, onde os círculos indicam os nós do mapa topológico e os arcos as interconexões entre os nós. P0, P1, P2 e PF indicam as poses do robô em cada nó.

4.1.3.2 Etapa de localização

A Figura 27 apresenta o fluxograma que ilustra o funcionamento da etapa de localização.

A primeira *thread* é responsável por chamar a função *videocap*, que deve receber a captura de imagens do robô pela câmera e fazer a contagem de quadros.

Para receber o sinal da câmera, um objeto *capture* da classe VideoCapture é criado. Esta é uma classe do OpenCV para captura

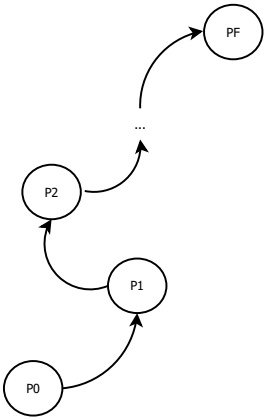


Figura 26 – Representação do mapa proposto

de vídeo por arquivos de vídeo, sequências de imagem ou câmeras. A classe fornece uma interface C++ para capturar vídeo de câmeras ou para ler arquivos de vídeo e sequências de imagem.

Após checar se a operação de receber os dados da câmera foi concluída com êxito, o quadro atual é armazenado na variável *frame*, que será usada para realizar a comparação na etapa de localização. Para um melhor controle, um contador é responsável por armazenar o número de cada quadro, identificando-o. Este dado será usado para posterior avaliação dos algoritmos propostos.

Após o algoritmo receber o primeiro quadro, indicando que o robô inicia sua trajetória, a *thread* responsável por chamar a função *matching*, que realiza a comparação para tentar inferir a localização do robô, é chamada.

A função *matching*, então, compara o quadro atual, recebido por *videocap* e identificado através da contagem de quadros com o conjunto de *keyframes*. Novamente, a comparação ocorre de forma idêntica à etapa de avaliação, descrita em 4.1.2. O maior número de correspondências entre o quadro atual e o conjunto de *keyframes* determina o *keyframe* mais próximo ao quadro atual, sendo assim, possível inferir a localização aproximada do robô na notação de nós com as respectivas

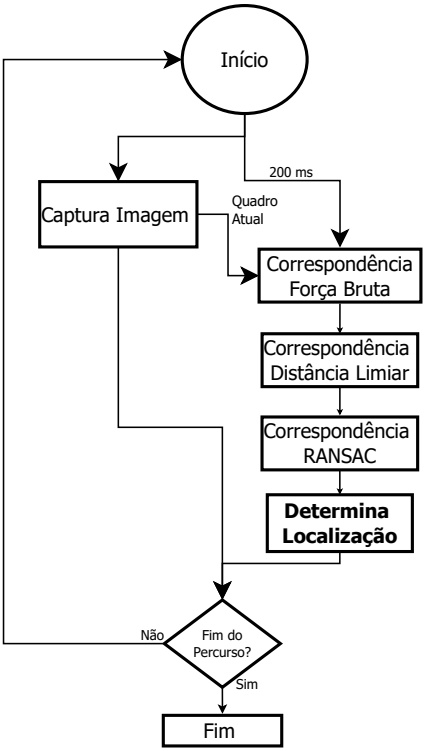


Figura 27 – Etapa de Localização

poses no mapa proposto. Estes dados são salvos juntamente com a informação de tempo em que foram obtidos para posterior avaliação dos algoritmos propostos.

A *thread* responsável pela captura das imagens executa sua função em um laço de repetição, que termina quando quando o robô não captura mais imagens, indicando que ele chegou ao fim da trajetória. O término da captura indica que novos quadros não estarão disponíveis a partir de então, logo, a função responsável por realizar as correspondência também não será chamada novamente.

4.1.4 Critérios de avaliação

A avaliação dos algoritmos propostos é feita comparando-se as poses obtidas na etapa de localização com o *groundtruth*. Sturm et al. (2012) propõem duas métricas: o erro relativo de pose (*Relative pose error* (RPE)) e o erro absoluto de trajetória (*Absolute trajectory error* (ATE)).

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

$$E_i := (Q_i^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}) \tag{4.2}$$

O erro de pose relativo mede a precisão local da trajetória ao longo de um intervalo de tempo fixo Δ. A partir de 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) propõem calcular o erro quadrático médio (RMSE) sobre todos os índices de tempo da componente translacional como:

$$RMSE(E_{1:n}, \Delta) := (\frac{1}{m} \sum_{i=1}^{m} \|\text{trans } E_i\|^2)^{\frac{1}{2}} \tag{4.3}$$

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) afirmem que a comparação por erros de translação é suficiente (visto que erros de rotação aparecem como erros de translação quando a câmera é movida).

Adicionalmente, a consistência global da trajetória estimada é uma métrica importante, podendo ser avaliada comparando-se as distâncias absolutas entre a trajetória estimada e a objetivo. Como ambas as trajetórias podem ser especificadas em quadros de coordenadas arbitrárias, elas precisam primeiro ser alinhadas. Isto pode ser conseguido em forma fechada 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:

$$F_i := Q_i^{-1} S P_i \tag{4.4}$$

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

$$RMSE(F_{1:n}) := (\frac{1}{n}\sum_{i=1}^{n}\|\text{trans } F_i\|^2)^{\frac{1}{2}} \tag{4.5}$$

Para essas avaliações, assume-se que tem-se uma sequência de poses a partir da trajetória estimada P1,. . . , Pn ∈ SE (3) e da trajetória verdadeira do solo Q1,. . . , Qn ∈ SE (3). Para simplificar a notação, será assumido que as sequências são sincronizadas no tempo, igualmente amostradas, e ambas têm comprimento n.

Na prática, independentemente do descritor 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.

Além disso, serão analisados os números de *keyframes* encontrados por cada descritor, bem como os respectivos impactos nas tarefas de localização e mapeamento.

4.2 SISTEMA DE REGISTRO 3D

Outro objetivo deste trabalho é o desenvolvimento de um sistema que possa construir uma representação 3D do ambiente em que o robô navega. Um limitante deste etapa é que o sistema desenvolvido é computacionalmente caro. Portanto, como o computador em que os testes acontecem não foi construído para este propósito, decidiu-se aplicá-lo a um espaço menor $(2,92m \times 3,18m)$. O algoritmo proposto recolhe imagens produzidas por um Kinect acoplado a um robô Pioneer em sua área de trabalho. As imagens RGB e de profundidade capturadas pelo Kinect são usadas para gerar nuvens de pontos. O algoritmo, então, busca encontrar uma transformação relativa (rotações/translações) em que cada nuvem se encaixe na anterior, formando assim um modelo global que representa o cenário percorrido pelo robô, respeitando as restrições de profundidade Kinect e de movimento do robô.

A primeira etapa do sistema proposto é a de aquisição de dados, em que o Kinect captura imagens do ambiente de trabalho do robô que são transformadas em nuvens de pontos. Essas nuvens de pontos recém adquiridas são processadas para dar inicio à etapa de reconstrução, que inicialmente consiste em encontrar uma transformação relativa que pré-alinha as nuvens às suas anteriores. O objetivo desta etapa é fornecer

um ”palpite inicial” para o algoritmo ICP, facilitando sua convergência. Por fim, através do algoritmo ICP, encontra-se uma transformação relativa para alinhar as nuvens e realizar a reconstrução 3D na forma de registro, conforme explicado em 2.2.3.

4.2.1 Recursos

A captura das imagens foi realizada através de um Microsoft Kinect acoplado a um robô Pioneer e a um computador portátil com processador Intel i7 2.90 GHz e 4 GB de memória RAM. O algoritmo para reconstrução 3D foi construído em C++ com o auxílio da biblioteca *Point Cloud Library* (PCL) (RUSU; COUSINS, 2011) em conjunto com o OpenNI no sistema operacional Linux Ubuntu 14.04.

4.2.2 Estrutura

A estrutura do software de reconstrução 3D é descrita a seguir.

4.2.2.1 Aquisição de nuvens de pontos

A PCL possui uma interface para comunicar com o Kinect e transformar as imagens em uma nuvem de pontos. A função *run()* de *SimpleOpenNIViewer* é responsável por criar uma nova interface *OpenNIGrabber*, que por sua vez realiza a tarefa de capturar as nuvens de pontos que são salvas através do comando *pcl::io::savePCDFileASCII (ss.str(), *outCld)*, onde *ss.sr()* indica o nome que a nuvem será salva e **outCld* identifica o objeto nuvem, que é criado para receber as nuvens de pontos provenientes do Kinect.

4.2.2.2 Algoritmo de registro

O fluxograma apresentado pela Figura 28 sumariza as etapas necessárias para reconstrução 3D na forma de registro após obtenção das nuvens de pontos.

Dois dos principais problemas enfrentados quando se tenta processar nuvens de pontos são ruído e densidade excessivos. O primeiro faz com que os algoritmos não interpretem corretamente os dados e produzam resultados incorretos ou imprecisos, enquanto o último faz

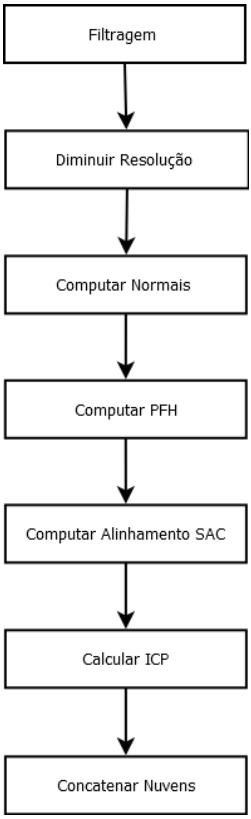


Figura 28 – Fluxograma de funcionamento do algoritmo de reconstrução 3D após captura das nuvens

com que os algoritmos levem tempo elevado para completar sua operação (MARTINEZ; FERNÁNDEZ, 2013). Portanto, para o processo de reconstrução, reduzir a quantidade de ruído ou *outliers* e a densidade de pontos sem perder informações valiosas é fundamental.

A primeira etapa de pré-processamento após adquirir as nuvens é a filtragem. As nuvens capturadas possuem ruídos, dados indesejados na forma de *NaN*(representando um valor de tipo de dados numérico que representa um valor indefinido ou irrepresentável) e dados esparsos. Isto complica a estimativa de características locais de nuvens de pontos, tais como extração de normais às superfícies ou mudanças de curvatura, levando a valores errôneos, o que por sua vez podem causar falhas no processo de registro. Portanto, esses erros devem ser minimizados a fim de que as nuvens possam ser usadas nos processos de alinhamento e reconstrução.

Alguns destes *outliers* podem ser filtrados realizando uma análise estatística em cada ponto de vizinhança, eliminando aqueles que não atendem a um determinado critério. A implementação de remoção esparsa em PCL baseia-se na computação da distribuição de distâncias entre pontos e vizinhos no conjunto de dados de entrada (RUSU, 2010) (RUSU; COUSINS, 2011). Para cada ponto, calcula-se a distância média dele a todos os seus vizinhos. Assumindo que a distribuição resultante é gaussiana com uma média e um desvio padrão, todos os pontos cujas distâncias médias estão fora de um intervalo definido pela média de distâncias globais e desvio padrão podem ser considerados como *outliers* e eliminados do conjunto de dados (RUSU, 2010).

Um exemplo de remoção de ruído é apresentado na Figura 29, que mostra os efeitos da análise e remoção esparsa: o conjunto de dados original é mostrado à esquerda, enquanto o resultante à direita.

A segunda etapa no processo de pré-processamento é a subamostragem das *point clouds*, processo também conhecido como *downsampling*. As nuvens recém adquiridas possuem cerca de 307200 pontos. Esse número excessivo de pontos torna o processamento extremamente lento. O desafio desta etapa é a diminuição desses pontos mantendo as propriedades básicas e a estrutura da nuvem de pontos.

O sistema proposto utiliza uma abordagem baseada em *voxel grids*. Um *voxel* funciona como uma caixa representada por uma grade regular no espaço tridimensional que substitui todos os pontos contidos dentro dele pelo centroide da subnuvem (MARTINEZ; FERNÁNDEZ, 2013). Tal como acontece com os pixeis de um mapa de bits, os próprios *voxels* não têm normalmente a sua posição (as suas coordenadas) explicitamente codificadas juntamente com os respectivos valores. Em

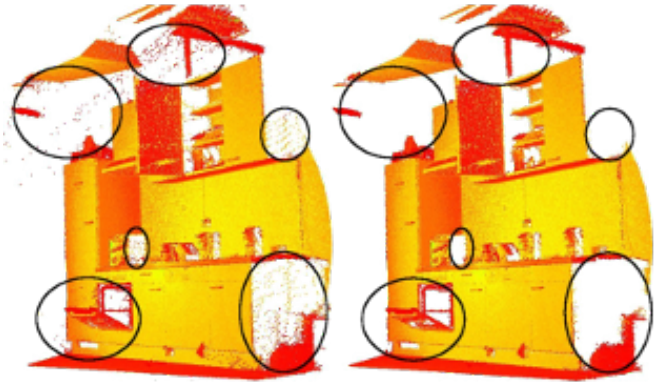


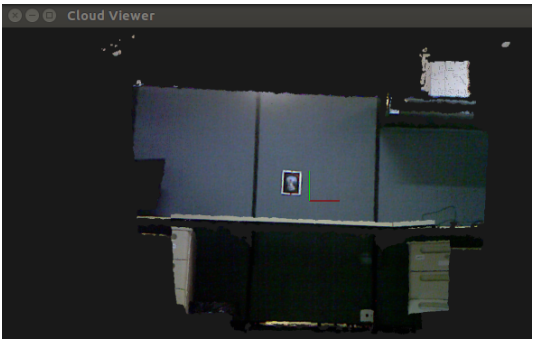
Figura 29 – Exemplo de filtragem de nuvens por análise estatística (RUSU, 2010)

vez disso, a posição de um *voxel* é inferida com base na sua posição em relação a outros *voxels* (isto é, a sua posição na estrutura de dados que constitui uma única imagem volumétrica) (RUSU, 2010). O tamanho de cada *voxel* pode ser especificado por $vox\_grid2.setLeafSize\ ()$, onde vox_grid2 é o objeto *voxel grid*, e $VOXEL\_GRID\_SIZE$ define o tamanho da grade.

Um problema importante na descrição da geometria da superfície é primeiro inferir sua orientação em um sistema de coordenadas, ou seja, estimar sua normal. Normais às superfícies são propriedades geométricas amplamente utilizadas em muitas áreas, como aplicações de computação gráfica, para aplicar as fontes de luz corretas que geram sombreamentos e outros efeitos visuais (RUSU, 2010).

Klasing et al. (2009) afirmam que existem muitos métodos de estimação de normais. O método utilizado, implementado pela biblioteca PCL, é formulado de forma que o problema de determinar o normal a um ponto na superfície é aproximado pelo problema de estimar o normal de um plano tangente à superfície, que por sua vez torna-se um problema de estimação de ajuste do plano de mínimos quadrados (RUSU, 2010).

Segundo Rusu (2010), a solução para estimar o normal de superfície é, portanto, reduzida a uma análise dos autovetores e autova-



(a) Nuvem filtrada

(b) Nuvem filtrada após subamostragem

Figura 30 – Etapa de *downsampling* com *voxel grids*

lores (ou PCA - Análise de Componentes Principais) de uma matriz de covariância criada a partir dos vizinhos mais próximos do ponto de consulta. Mais especificamente, para cada ponto $\boldsymbol{p}_i$, monta-se a matriz de covariância $\mathcal{C}$ da seguinte maneira:

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ \mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j},\ j \in \{0, 1, 2\} \quad (4.6)$$

Onde k é o número de vizinhos pontuais considerados na vizinhança de $\boldsymbol{p}_i$, $\overline{\boldsymbol{p}}$ representa o centroide 3D dos vizinhos mais próximos, λ_j é o j-ésimo autovalor da matriz de covariância , e $\vec{\mathrm{v}_j}$ o j-ésimo vetor próprio.

Em geral, uma vez que não há nenhuma maneira matemática de

resolver o sinal da normal, sua orientação calculada via PCA é ambígua e não consistentemente orientada sobre um conjunto de dados de nuvem de pontos. A solução para este problema é trivial se o ponto de vista v_p é conhecido. Para orientar todas as normais $\vec{\boldsymbol{n}}_i$ consistentemente para o ponto de vista, elas precisam satisfazer a equação (RUSU, 2010):

$$\vec{\boldsymbol{n}}_i \cdot (\mathsf{v}_p - \boldsymbol{p}_i) > 0 \tag{4.7}$$

Rusu (2010) afirma que a escolha do correto fator de escala r (*pcl::Feature::setRadiusSearch*) é de extrema importância. Um fator de escala razoável deve exibir normais de superfície estimadas aproximadamente perpendiculares para superfícies planares e as bordas pequenas visíveis. A parte esquerda da Figura 31 representa um fator de escala razoavelmente bem escolhido, com as normais de superfície estimadas aproximadamente perpendiculares para as duas superfícies planares. Se o fator de escala for muito grande (parte direita) e, assim, o conjunto de vizinhos do ponto for maior, cobrindo pontos das superfícies adjacentes, as representações das características do ponto estimado ficam distorcidas, com normais de superfície giradas nas bordas das duas superfícies planares, ocasionando bordas manchadas e detalhes finos suprimidos.

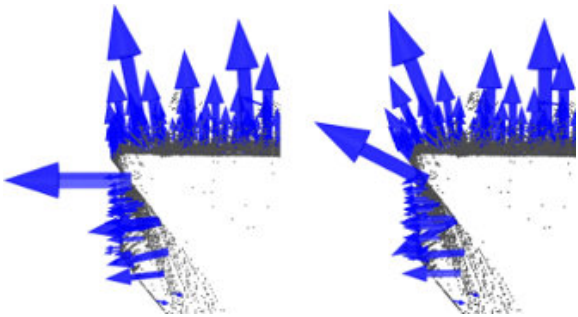


Figura 31 – Exemplo de normais de superfície estimadas para um subconjunto de pontos usando um bom fator de escala (à esquerda) e um grande (inadequado) fator de escala (direita). (RUSU, 2010)

Neste projeto, as normais são estimadas através da função *getNormals(PointCloud<PointXYZRGB>::Ptr incloud)*, que recebe a nuvem objetivo, isto é, a nuvem da qual as normais devem ser computadas e retorna o conjunto de normais de cada nuvem. Para usar a classe *NormalEstimation* da PCL para calcular as normais de superfície foram especificados a nuvem do pontos de entrada, a árvore kd a ser usada na

busca de pontos vizinhos e o raio que define a vizinhança de cada ponto. Em seguida, calcula-se as superfícies normais, que são armazenadas em uma variável para uso posterior. A escolha dos parâmetros r e tamanho da arvore foi feita empiricamente, atribuindo valores e observando resultados.

A Figura 32 apresenta as normais encontradas para uma das nuvens obtidas no percurso do robô.

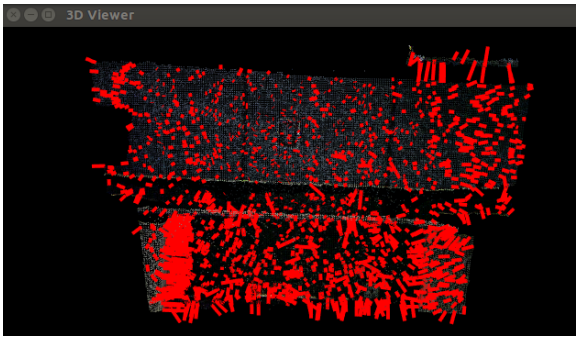


Figura 32 – Normais obtidas em uma *point cloud* após etapa de *downsampling*

As características locais necessárias para realizar o alinhamento inicial SAC são computadas através dos descritores PFH e FPFH, que foram propostos justamente para a tarefa de registro (RUSU et al., 2008). A função *getfeatures()* recebe como parâmetros as nuvens e suas respectivas normais e retorna as características computadas em cada nuvem.

A implementação da PFH padrão da PCL usa 5 subdivisões (por exemplo, cada um dos quatro valores de característica usará este número de subdivisões de seu intervalo de valores) e não inclui as distâncias d (embora o método *computePairFeatures* possa ser chamado pelo usuário para obter as distâncias também, se desejado). Estes são armazenados em uma variável do tipo *pcl::PFHSignature125*.

Já a implementação FPFH padrão usa 11 subdivisões de compartimentos (por exemplo, cada um dos quatro valores de característica usará esses muitos compartimentos de seu intervalo de valores) e um esquema em que os histogramas das características são computados separadamente e concatenados. Estes dados são armazenados em variáveis do tipo *pcl::FPFHSignature33*.

Em seguida, o alinhamento inicial é definido. O método utilizado usa um modelo como entrada (uma nuvem de pontos) e o alinha

à nuvem de destino que foi especificada chamando *setInputTarget ()*. Ele funciona definindo o modelo fornecido como a nuvem de origem do algoritmo SAC-IA e, em seguida, chamando seu método *align ()* para alinhar a origem ao destino. A função responsável por encontrar a transformada SAC-IA recebe duas nuvens e suas respectivas características, retornando a transformada desejada, responsável pelo alinhamento inicial.

Para utilizar a implementação do SAC-IA da PCL, são definidos os parâmetros de distâncias mínimas entre as amostras e a máxima distância entre correspondências.

A Figura 33 apresenta o alinhamento inicial entre duas nuvens consecutivas obtidas usando o descritor FPFH.

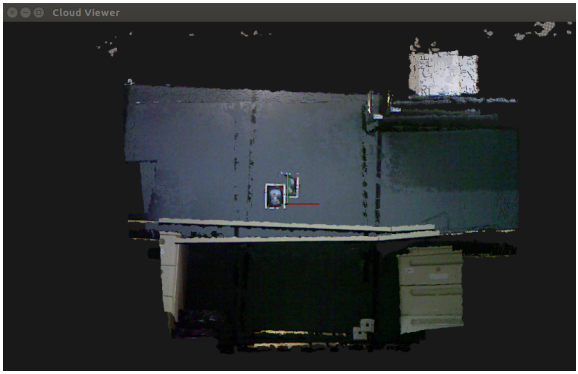


Figura 33 – Alinhamento Inicial SAC-IA entre duas nuvens

O algoritmo proposto calcula essa transformada inicial para o primeiro par de nuvens. O próximo passo é um alinhamento fino, realizado pelo algoritmo ICP em cima das nuvens pré-alinhadas. Após encontrar a transformação que melhor alinha as duas primeiras nuvens, a matriz de transformação é aplicada na segunda nuvem que servirá de modelo pra um novo alinhamento inicial SAC-IA com a seguinte nuvem. Aplica-se novamente o ICP para encontrar a transformada entre a nuvem dois - já alinhada com a primeira - e a nuvem três. O processo se repete até a última nuvem. Este trecho do programa, é apresentado no fluxograma da Figura 34.

Para o cálculo do ICP, definiu-se o número máximo de iterações como 100, a máxima distância de correspondência como 0,05, equivalente a 5 centímetros, e o máximo *epsilon* da transformada como 1e-9. O máximo *epsilon* é um dos critérios de parada do algoritmo ICP, em

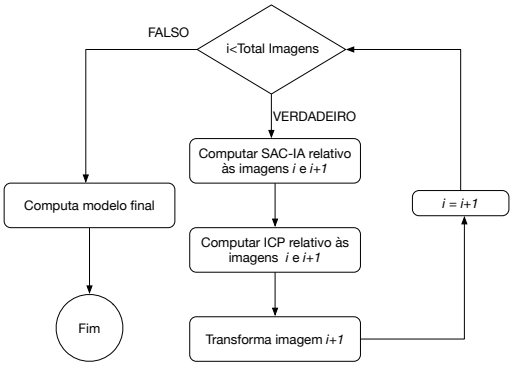


Figura 34 – Fluxograma de funcionamento da etapa de alinhamento

que o *epsilon* (diferença entre a transformação anterior e a transformação estimada atual) é menor do que um valor imposto pelo usuário (via *setTransformationEpsilon*) .

Após encontrar e aplicar as transformadas que melhor alinham as nuvens obtidas segundo o algoritmo proposto, as nuvens de pontos são concatenadas a fim de construir um modelo global do ambiente de trabalho do robô.

O alinhamento final entre as duas nuvens apresentadas na Figura 33 obtido pelo ICP é apresentado pela Figura 35.

Figura 35 – Alinhamento final ICP entre duas nuvens após SAC-IA

4.2.3 Critérios de avaliação

Serão avaliados os resultados qualitativos e quantitativos do registro 3D a partir dos dados obtidos. Serão comparados o desempenho dos descritores PFH e FPFH, além da influência do alinhamento inicial SAC-IA.

A avaliação qualitativa consiste na apresentação e observação dos modelos obtidos e sua verossimilhança com a realidade. Ou seja, se através do modelo obtido é possível entender a cena.

O objetivo do SAC-IA é fornecer um alinhamento inicial para o algoritmo ICP. Desta forma, serão analisadas a convergência do ICP através do tempo com e sem alinhamento inicial.

Além disso, será estudada a influência dos descritores PFH e FPFH para alinhamento inicial SAC-IA e seu impacto no ICP, utilizando como métricas o tempo de extração das características, de convergência ICP após alinhamento inicial para cada descritor e do processo por completo, além da análise do resultado final.

5 AVALIAÇÃO DOS RESULTADOS

Este capítulo visa apresentar as avaliações dos sistemas propostos com base nos critérios descritos no capítulo anterior. Inicialmente, os descritores BRIEF, BRISK, FREAK, ORB, SIFT e SURF são avaliados para verificar sua aplicabilidade no problema de localização e mapeamento robótico no contexto proposto por este trabalho. Em seguida, são apresentadas as avaliações dos sistemas de mapeamento e localização, e registro 3D.

5.1 AVALIAÇÃO DE DESCRITORES

Esta seção apresenta uma comparação de BRIEF, BRISK, FREAK, ORB, SIFT e SURF com base nos critérios de avaliação descritos na seção 4.1.2. Cada subseção apresenta um teste que avalia características relevantes para o sistema de localização e mapeamento proposto.

5.1.1 Teste local

Neste teste, os quadros aleatórios integrantes da trajetória do robô foram selecionados pelos critérios propostos para realizar o processo de correspondência com seus dez vizinhos anteriores e próximos. Os testes mostraram resultados semelhantes. As Figuras 36 e 37 sumarizam o comportamento dos descritores.

Os resultados mostram que a maioria dos descritores apresentou o comportamento desejado, especialmente BRISK, ORB, SIFT e SURF, com todas as suas curvas se aproximando melhor a uma distribuição normal. FREAK, por outro lado, não possui um bom desempenho neste teste, pois em algumas situações apresenta distribuições diferentes da distribuição normal, com quadros distantes obtendo números de correspondências similares aos quadros mais próximos ao quadro alvo.

5.1.2 Teste global

As Tabelas 1 e 2 apresentam os resultados do teste global, onde **Q** representa o número do quadro e **C** o número de correspondências. Para

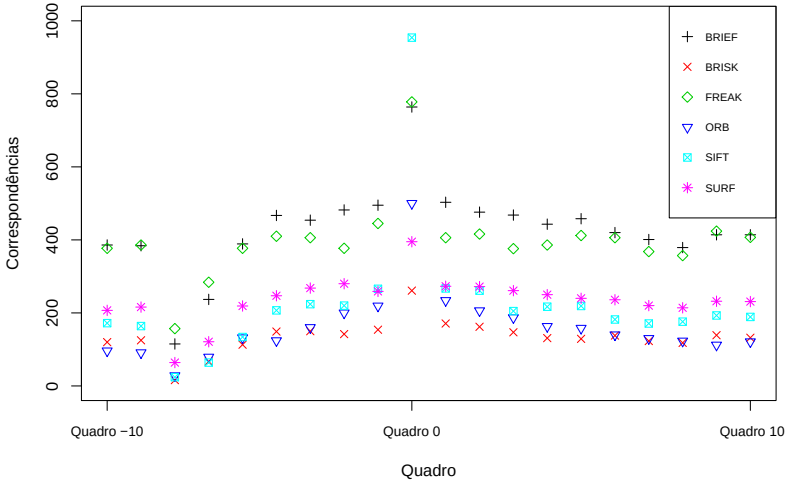


Figura 36 – Teste local - Quadro de referência 714

cada descritor, o quadro de destino foi comparado com todo o conjunto de dados e os cinco quadros com o maior número de correspondências foram retornados.

Tabela 1 – Teste global - Quadro de referência: 714

	Q/C	Q/C	Q/C	Q/C	Q/C
BRIEF	715/503	713/495	712/482	716/476	717/468
BRISK	715/171	716/162	713/154	711/150	710/149
FREAK	166/138	236/130	2147/130	168/124	2803/124
ORB	715/234	713/219	716/206	712/200	717/187
SIFT	715/267	713/266	716/261	704/246	711/224
SURF	712/280	715/273	716/272	711/278	717/261

Todos os descritores menos FREAK apresentam o resultado desejado neste teste, encontrando quadros vizinhos ao quadro de referência. Esse tipo de teste é relevante no contexto do problema de localização global, em que é fundamental que o robô consiga distinguir diferentes regiões integrantes de sua trajetória.

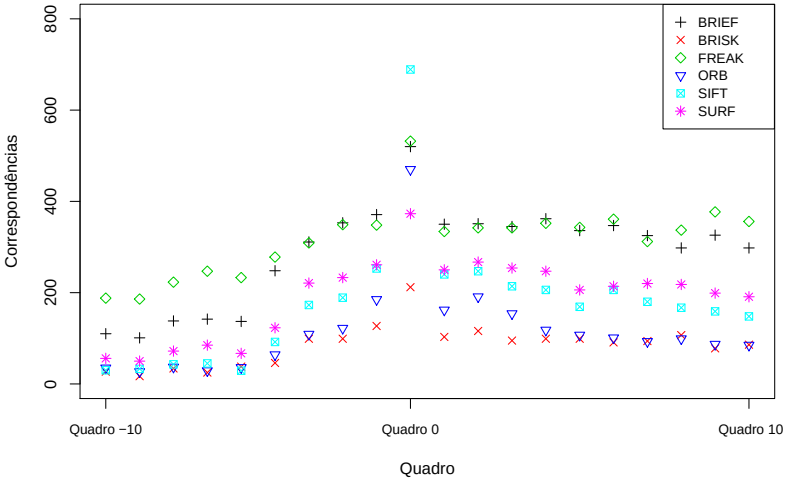


Figura 37 – Teste local - Quadro de referência 2651

Tabela 2 – Teste global - Quadro de referência: 2651

	Q/C	Q/C	Q/C	Q/C	Q/C
BRIEF	2650/371	2655/363	2649/353	2653/351	2652/352
BRISK	2650/127	2653/116	2659/107	2652/103	2648/99
FREAK	2802/197	2508/188	2517/186	1930/185	2519/179
ORB	2653/191	2650/185	2652/154	2649/122	2655/118
SIFT	2650/253	2653/247	2652/240	2654/214	2655/206
SURF	2653/267	2650/261	2654/254	2652/250	2655/247

5.1.3 Teste de velocidade de correspondência

O teste de velocidade de correspondência apresenta o tempo de correspondência médio obtido a partir do teste global. Este teste é realizado medindo a velocidade média de correspondência do quadro de destino com o restante da trajetória para cada descritor. Os resultados estão presentes na Tabela 3. O tempo médio de correspondência entre quadros é apresentado em milissegundos (ms).

A combinação mais rápida é obtida por BRISK. Os resultados de

Tabela 3 – Velocidade de correspondência (ms)

	TUM RGB-D
BRIEF	62,14
BRISK	19,48
FREAK	82,86
ORB	34,71
SIFT	152,45
SURF	20,19

SURF também são impressionantes, sendo um dos mais velozes neste critério, e mais rápido que outros descritores que empregam a distância de Hamming, como FREAK, BRIEF.

5.1.4 Teste de dissimilaridade

A Tabela 4 apresenta os resultados do teste de dissimilaridade. Usando os dados do teste local, esse teste visa verificar o quão distintivo cada descritor é em respeito a quadros similares. Logo, calcula-se a distância euclidiana média entre o número de *keypoints* do quadro-objetivo que são encontrados nos quadros em sua vizinhança. Uma pequena distância mostra um alto grau de similaridade e uma grande demonstra um baixo grau.

Tabela 4 – Teste de dissimilaridade

	Distância Euclidiana
BRIEF	292,7262
BRISK	140,3895
FREAK	247,8211
ORB	385,0895
SIFT	613,5436
SURF	182,8157

SIFT apresentou melhor desempenho neste teste quando comparado aos outros descritores, sendo mais de três vezes mais distintivo que SURF e BRISK. Os resultados de ORB também são razoáveis, sendo o melhor entre os descritores binários.

Desta forma, analisando os resultados, verificou-se que todos os descritores analisados, exceto FREAK, atendem às características ne-

cessárias para o sistema de mapeamento e localização na forma que foi proposta.

5.2 ALGORITMOS DE LOCALIZAÇÃO E MAPEAMENTO

Esta seção apresenta os resultados da avaliação dos algoritmos de localização e mapeamento propostos.

5.2.1 Número de *keyframes*

A Tabela 5 apresenta o número de *keyframes* encontrados por cada descritor usando a metodologia descrita em 4.1.4.

Tabela 5 – Número de *Keyframes*

	Número de *Keyframes*
BRIEF	51
BRISK	44
ORB	55
SIFT	84
SURF	47

O descritor SIFT encontrou um maior número de *keyframes*. O número de *keyframes* representa, em geral, o grau de detalhe dos mapas topológicos, em que muitos nós representam um mapa mais detalhado, com mais pontos para localização, o que pode contribuir para uma localização mais precisa. Por outro lado, um alto número de nós pode implicar em um maior tempo de correspondência na etapa de localização, fazendo com que o robô se localize menos vez durante sua trajetória, causando assim o efeito inverso do desejado, uma localização em pontos mais esparsos e, portanto, menos precisa durante toda trajetória.

5.2.2 *Relative Pose Error* (RPE)

O RPE mede a diferença entre o movimento estimado e o movimento real, podendo ser usado para avaliar um sistema de odometria visual ou a precisão em fechamentos de loop de sistemas SLAM, o que é especialmente útil se apenas as relações esparsas e relativas estão disponíveis como *groundtruth*. Além disso, o RPE pode ser usado para

avaliar o erro global de uma trajetória por média em todos os intervalos de tempo possíveis (STURM et al., 2012), como no caso apresentado por este trabalho.

A Tabela 6 apresenta os resultados do critério RPE em metros (m), onde os erros RMSE e máximo são apresentados como RMSE e MAX. A letra T antes de RMSE e MAX é referente aos erros translacionais.

Tabela 6 – RPE (m)

	TRMSE	TMAX
BRIEF	0,40847	0,539263
BRISK	0,16473	0,45642
ORB	0,24339	0,292293
SIFT	0,653462	2,317167
SURF	0,20218	0,84816

Verificou-se que todos descritores foram capazes de localizar o robô com razoável precisão, destacando-se os descritores BRISK, ORB e SURF, com erros translacionais relativamente mais baixos.

O descritor SIFT obteve um erro comparativamente maior que os outros. Este fato deve-se ao alto número de *keyframes* encontrados aliado a um alto tempo de correspondência deste descritor.

5.2.3 *Absolute Trajectory Error* (ATE)

Em vez de avaliar diferenças de poses relativas, o ATE primeiro alinha as duas trajetórias e, em seguida, avalia diretamente as diferenças de pose absolutas. Este método é adequado para a avaliação de sistemas SLAM visuais, mas requer que a verdade absoluta (*ground-truth*) das poses na trajetória estejam disponíveis.

O teste *Absolute trajectory Error* calcula os erros RMSE e máximo conforme metodologia apresentada na seção 4.1.4. A Tabela 7 apresenta os resultados do ATE em metros.

Os resultados desta métrica ratificam as conclusões observadas através do RPE, com os descritores BRISK, SURF e ORB obtendo erros comparativamente mais baixos.

Tabela 7 – ATE (m)

	RMSE	MAX
BRIEF	0,39074	0,539182
BRISK	0,15346	0,45761
ORB	0,17840	0,284688
SIFT	0,629034	1,838244
SURF	0,19432	0,83817

5.2.4 Comparação com estado da arte

Nesta subseção, os sistemas desenvolvidos foram comparados com os sistemas desenvolvidos por Endres et al. (2014) (processador Intel Core i7 3,40GHz, e placa de vídeo nVidia GeForce GTX 570 graphics), Concha e Civera (2017) e Whelan et al. (2016) (processador 3,5 GHz Intel Core i7 e 8 GB de memória RAM para os dois últimos), que tratam-se do estado-da-arte em sistemas RGB-D SLAM em tempo e precisão, superando todas as linhas de base anteriores consultadas.

A Tabela 8 apresenta os resultados do erro absoluto de trajetória em centímetros para a sequência fr1/desk1, que foi a única avaliada em todos os sistemas. Trata-se de uma sequência em que o Kinect é manualmente conduzido um ambiente com duas mesas.

Tabela 8 – ATE RMSE (cm)

	fr1/desk
BRIEF	7,9
BRISK	6,5
ORB	6,4
SIFT	8,6
SURF	7,5
Endres et al. (2014)	2,6
Concha e Civera (2017)	2,7
Whelan et al. (2016)	2,0

Apesar de existirem técnicas mais precisas, os descritores testados para o problema proposto foram capazes de resolvê-lo de forma confiável, com erros toleráveis para esta proposta. É importante ressaltar que os sistemas no estado da arte fazem uso de fusão sensorial, utilizando múltiplas leituras de diferentes sensores para encontrar uma

localização mais precisa, além de sistemas computacionais mais poderosos. No sistema desenvolvido não é necessário nenhum outro sensor além de uma simples câmera, o que é extremamente interessante em aplicações de custo reduzido.

Embora uma maior dissimilaridade contribua para um mapa mais detalhado, verificou-se que a velocidade de correspondência entre quadros é um fator mais preponderante na tarefa de localização, visto que o descritores SURF, ORB e BRISK foram os que obtiveram menores erros durante sua trajetória.

5.2.5 Análise de requisitos

5.2.5.1 Requisitos funcionais

RF-1. Escopo do sistema de localização e mapeamento

O sistema proposto é capaz de localizar o robô Activ Media Pioneer nos mapas criados durante toda sua trajetória. Portanto, este requisito foi atendido.

RF-2. Composição do mapa

Foram propostos mapas topológicos em que o problema de fechamento de loop é analisado. Logo, os mapas propostos contêm características suficientes para localizar o robô durante sua trajetória, estando atentos ao fato de que o robô pode retornar a uma localização previamente visitada.

RF-3. Compatibilidade do sistema de localização com sistemas robóticos

Os sistemas desenvolvidos utilizaram a linguagem C++ e a biblioteca OpenCV, ambas possuem interface com ROS, que fornece bibliotecas e ferramentas para ajudar os desenvolvedores de software a criar aplicativos para robôs. Desta forma, este requisito também foi atendido.

5.2.5.2 Requisitos não-funcionais

RNF-1. Tempo de setup

Os erros relativos ao tempo de inicialização da câmera não afetam significativamente a localização do robô nos sistemas propostos, visto que mesmo os erros máximos são toleráveis para esta proposta.

RNF-2. Tempo de correspondência

O descritor SIFT possui, de acordo com os testes realizados, o maior tempo médio de correspondência entre quadros, porém, ainda assim, trata-se de um tempo suficientemente curto visto que a velocidade média do robô é de 0,261 m/s. Portanto, o tempo de correspondência entre imagens foi considerado suficientemente satisfatório.

RNF-3. Marcadores naturais

Através da utilização de descritores de características locais, não necessitou-se de marcações artificias. Para realizar sua tarefa de mapeamento e localização, foram utilizados somente marcações naturais encontradas na trajetória do robô.

RNF-4. Confiabilidade da localização

Apesar de suas limitações, as técnicas propostas são capazes de localizar o robô de forma absoluta, ou seja, sem qualquer necessidade de saber a pose do robô no instante anterior. Por comparar a imagem obtida no instante atual com todo o conjunto de *keyframes*, o algoritmo desenvolvido também é capaz de resolver o problema do robô sequestrado e do robô despertado. Logo, este requisito também foi atingido.

5.3 REGISTRO 3D

A Tabela 9 apresenta o tempo consumido pelo algoritmo ICP sem alinhamento inicial e a Figura 38 apresenta o resultado final desta operação em um conjunto de 46 nuvens de pontos que representam o espaço de trabalho do robô.

Tabela 9 – Tempos de convergência sem SAC-IA (s)

	Tempo (s)
ICP	5380

O resultado possui uma série de distorções causadas pelo fato de algoritmo ter de encontrar transformações relativas entre nuvens que satisfaçam ambos os critérios parada com um número máximo de iterações, além do fato de estar sujeito a cair em mínimos locais.

As Tabelas 10 e 11 apresentam os tempos de extração das características PFH e FPFH, dos alinhamentos inicial SAC-IA e final ICP para cada combinação de algoritmos referente às 46 nuvens de pontos. As Figuras 39 e 40 apresentam os respectivos resultados finais.

Verifica-se que o resultado obtido utilizando o descritor PFH é satisfatório, principalmente pelo fato da região modelada conter um alto grau de detalhes e objetos, além dos diferentes pontos de vista

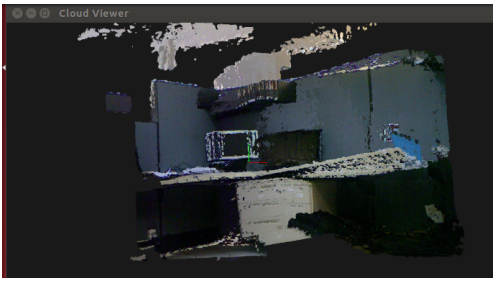


Figura 38 – Alinhamento final ICP

Tabela 10 – Tempos de convergência com SAC-IA e PFH (s)

	Tempo (s)
PFH	593533
SAC-IA	5654
ICP	3440
TEMPO TOTAL	602690

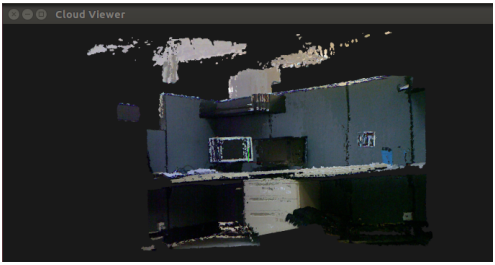


Figura 39 – Alinhamento final ICP com SAC-IA e PFH

devido às mudanças de direções do robô, o que dificulta o processo de registro. A operação total consumiu mais de 167 horas, ou seja, mais de seis dias. É importante salientar que somente a extração das características foi responsável por cerca de 164 horas.

Utilizando o descritor FPFH, o tempo total consumido pelo algoritmo foi consideravelmente alto, ultrapassando 9 horas, porém muito menor que o tempo consumido utilizando PFH, com resultado final bastante similar.

Assim, observa-se que a relação de troca entre o tempo de extração de características, tempos para alinhamento inicial e final e resul-

Tabela 11 – Tempos de convergência com SAC-IA e FPFH (s)

	Tempo (s)
FPFH	22787
SAC-IA	6359
ICP	4032
TEMPO TOTAL	33237

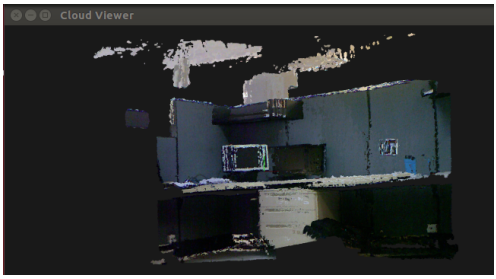


Figura 40 – Alinhamento final ICP com SAC-IA e FPFH

tado final utilizando FPFH é melhor que PFH, pois ambos obtiveram resultados finais visualmente similares. A utilização de um ”palpite inicial” pelo ICP fornecida pelo SAC-IA demonstrou-se útil no ponto de vista de criar modelos mais similares com a realidade, um problema encontrado foi o alto tempo que as operações que compõem esse alinhamento inicial demandam.

5.3.1 Análise de requisitos

5.3.1.1 Requisitos funcionais

RF-1. Escopo do sistema de reconstrução

O sistema desenvolvido cria nuvens de pontos através de imagens percebidas por um Microsoft Kinect acoplado a um robô Pioneer enquanto ele percorre sua trajetória e reconstrói tridimensionalmente esse ambiente. Desta forma, este requisito foi atendido.

RF-2. Ambiente de trabalho

Os testes foram realizados enquanto o robô Pioneer percorre sua trajetória em seu ambiente de trabalho, respeitando, assim, este requisito funcional.

RF-3. Compatibilidade do sistema de registro com sistemas robóticos

Os sistemas desenvolvidos utilizaram a linguagem C++ e a biblioteca PCL, ambas possuem interface com ROS, que fornece bibliotecas e ferramentas para ajudar os desenvolvedores de software a criar aplicativos para robôs. Desta forma, este requisito também foi atendido.

5.3.1.2 Requisitos não-funcionais

RNF-1. Características do Kinect

Optou-se por realizar os testes em um ambiente com pouca amplitude devido ao fato do Kinect possuir limitações de distância em sua captura. Assim, foi possível minimizar esta limitação.

RNF-2. Velocidade de movimento do Kinect

O movimento do robô ocorreu de forma suave, de forma a evitar movimentos bruscos, rápidas velocidades, minimizando, assim, o efeito desta limitação no processo de registro.

RNF-3 e RNF-4 Confiabilidade da reconstrução e Informações

O modelo tridimensional obtido pelo sistema de registro é verossímil à realidade, possuindo informações de cor e profundidade, o que permite um fácil entendimento do ambiente de trabalho do robô.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÃO

Devido a seu caráter multidisciplinar, os problemas de localização e mapeamento robótico têm sido um constante alvo de pesquisadores das mais variadas áreas. A introdução de visão em um robô tem se mostrado uma alternativa válida, sendo capaz de obter um alto grau de informação com preços comparativamente baixos.

A utilização de técnicas exclusivamente baseadas em visão computacional nos problemas de localização e mapeamento robótico 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.

A avaliação demonstrou que, apesar de suas limitações, essas técnicas são capazes de localizar o robô de forma absoluta, ou seja, sem qualquer necessidade de saber a pose do robô no instante anterior. Por comparar a imagem obtida no instante atual com todo o conjunto de *keyframes*, o algoritmo desenvolvido também é capaz de resolver o problema do robô sequestrado, em que o robô sabe exatamente onde está localizado, mas de repente ele é transferido, ou "sequestrado", para outro local sem que o robô esteja ciente disso. O problema para o robô é detectar que ele foi sequestrado e descobrir qual é a sua nova localização, e também do robô que acaba de despertar, que é um caso especial da anterior, em que um robô autônomo é transportado para uma localização arbitrária e colocado em operação, e o robô deve localizar-se. Essas situações são comumente usadas para testar a capacidade de um robô para se recuperar de falhas catastróficas de localização.

Desta forma, os descritores SURF, ORB e BRISK foram os que obtiveram menores erros durante sua trajetória, destacando a importância da velocidade de correspondência entre quadros para uma localização mais precisa.

O sistema de mapeamento e localização proposto tem como principais desvantagens a necessidade de um sistema de captura externo capaz de obter o *groundtruth* da trajetória e de uma fase de treinamento que contenha possíveis poses do robô (pelo menos de forma aproximada) em sua etapa de navegação. As localizações obtidas nas fases de navegação demonstraram-se suficientemente precisas de acordo com a proposta deste trabalho, porém é interessante investigar sistemas mais precisos, principalmente os que fazem uso de múltiplos sensores, em-

pregando fusão temporal ou fusão multissensorial.

Modelar um ambiente de forma tridimensional permite obter um alto grau de detalhe, 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 necessitem 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.

A reconstrução 3D na forma de registro proposta neste trabalho apresentou resultados satisfatórios, criando modelos compreensíveis aos usuários, porém o alto custo computacional é um fator a ser levado em conta, principalmente em aplicações em que o robô percorre grandes trajetórias ou que um alto poder computacional não esteja disponível. Outro fator limitante desta técnica é que o uso de câmeras do tipo Kinect impõe algumas restrições, como as distância máxima e mínima que podem ser captadas pelo sensor e utilizadas para a criação de nuvens de pontos, que limitam o registro em ambientes em que o horizonte está distante.

A utilização do alinhamento inicial SAC-IA demonstrou-se útil, pois contribui para um processo de registro mais verossímil, porém aumenta demasiadamente o tempo do mesmo. É importante salientar que ainda assim a técnica demonstrou-se eficiente e pode ser usada sempre que o tempo não for um fator limitante ou quando se dispõe de um alto poder computacional.

Dentre os descritores usados para alinhamento inicial SAC-IA, a simplificação do PFH, FPFH demonstrou-se como boa alternativa, pois possui tempos de extração muito menores que PFH não afetando significantemente os alinhamentos inicial e final.

De modo geral, os sistemas propostos neste projeto demonstraram-se capazes de resolver os problemas para que foram desenvolvidos, permitindo com que um robô se localize durante sua trajetória e seja capaz de construir um modelo 3D de seu ambiente de trabalho. Verificou-se, também, que todos os requisitos funcionais e não-funcionais dos sistemas propostos foram atendidos.

6.2 TRABALHOS FUTUROS

Os aspectos observados que podem ser melhorados ou aprofundados com o objetivo de expandir este trabalho são:

- Os vários sensores usados para a tarefa de navegação robótica possuem suas limitações, que podem aparecer na forma de ruído, na quantidade e qualidade da informação disponíveis, etc. Seria interessante, portanto, estudar a integração do sistema de visão com outros sensores, levando em consideração múltiplas leituras, empregando fusão temporal ou fusão multissensor para aumentar o conteúdo de informação geral das entradas do robô;
- A reconstrução 3D na forma proposta é especialmente interessante para usuários remotos. Porém, seria recomendável estudar maneiras que essa representação pode se converter em um mapa para o robô;
- Realizar as etapas de mapeamento de localização simultaneamente;
- A representação 3D na forma de registro proposta neste trabalho tem como restrição o elevado tempo de processamento. Recomenda-se, portanto, o estudo de técnicas capazes de realizar esta tarefa em menor tempo, idealmente em tempo real enquanto o robô se move em seu ambiente;
- Investigar navegação baseada em visão sem etapa de treinamento;
- Substituir Kinect por par de câmeras convencionais;
- Por fim, sugere-se investigar a navegação sem mapas baseada em visão computacional.

REFERÊNCIAS

AGRAWAL, M.; KONOLIGE, K. Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS. In: **18th International Conference on Pattern Recognition (ICPR'06)**. IEEE, 2006. v. 3, p. 1063–1068. ISBN 0-7695-2521-0. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1699709>.

ALAHI, A.; ORTIZ, R.; VANDERGHEYNST, P. FREAK: Fast Retina Keypoint. In: **2012 IEEE Conference on Computer Vision and Pattern Recognition**. IEEE, 2012. p. 510–517. ISBN 978-1-4673-1228-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247715>.

ALEKSANDROVICH, Y. D. et al. Mobile robot navigation based on artificial landmarks with machine vision system. **World Applied Sciences Journal**, v. 24, n. 11, p. 1467–1472, 2013.

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. In: CITESEER. **CCIA**. [S.l.], 2008. p. 363–371.

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. **2013 IEEE International Conference on Image Processing**. [S.l.], 2013. p. 3652–3656.

BELLEKENS, B. et al. A survey of rigid 3d pointcloud registration algorithms. In: CITESEER. **Fourth International Conference on Ambient Computing, Applications, Services and Technologies**. [S.l.], 2014. p. 8–13.

BELLOTTO, N. et al. Appearance-based localization for mobile robots using digital zoom and visual compass. **Robotics and Autonomous Systems**, Elsevier, v. 56, n. 2, p. 143–156, 2008.

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.

BESL, P. J.; MCKAY, N. D. Method for registration of 3-d shapes. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Robotics-DL tentative**. [S.l.], 1992. p. 586–606.

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.

BONARINI, A. et al. Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. In: **In proceedings of IROS**. [S.l.: s.n.], 2006. v. 6.

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

BONNAL, E. P. 3d mapping of indoor environments using rgb-d kinect camera for robotic mobile application. Universitat Politècnica de Catalunya, 2010.

BRADSKI, G. et al. The opencv library. **Doctor Dobbs Journal**, M AND T PUBLISHING INC, v. 25, n. 11, p. 120–126, 2000.

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

CHANG, C. K.; SIAGIAN, C.; ITTI, L. Mobile robot vision navigation & localization using gist and saliency. **IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings**, p. 4147–4154, 2010. ISSN 2153-0858.

CHEN, Y.; MEDIONI, G. Object modelling by registration of multiple range images. **Image and vision computing**, Elsevier, v. 10, n. 3, p. 145–155, 1992.

CHI, W. et al. A Vision-based Mobile Robot Localization Method. **Proceeding of the IEEE International Conference on Robotics and Biomimetics**, n. December, p. 2703–2708, 2013.

CLEMONS, J. Sift: Scale invariant feature transform by david lowe. **Lecture, Opened**, v. 2, 2014.

CONCHA, A.; CIVERA, J. Rgbdtam: A cost-effective and accurate rgb-d tracking and mapping system. **arXiv preprint arXiv:1703.00754**, 2017.

COUTSIAS, E. A. et al. A kinematic view of loop closure. **Journal of computational chemistry**, Wiley Online Library, v. 25, n. 4, p. 510–528, 2004.

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.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. IEEE, 2005. v. 1, p. 886–893. ISBN 0-7695-2372-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467360>.

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. An evaluation of the rgb-d slam system. In: IEEE. **Robotics and Automation (ICRA), 2012 IEEE International Conference on**. [S.l.], 2012. p. 1691–1696.

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, oct 2012. ISSN 0162-8828. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6112779>.

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. **Communications of the ACM**, ACM, v. 24, n. 6, p. 381–395, 1981.

FOX, D.; BURGARD, W.; THRUN, S. Markov Localization for Mobile Robots in Dynamic Environments. **Journal of Artificial Intelligence Research**, v. 11, p. 391–427, 1999. ISSN 1076 - 9757.

FREEMAN, W. T.; ADELSON, E. H. The design and use of steerable filters. **IEEE Transactions on Pattern analysis and machine intelligence**, v. 13, n. 9, p. 891–906, 1991.

GARCIA-FIDALGO, E.; ORTIZ, A. Probabilistic appearance-based mapping and localization using visual features. In: SPRINGER. **Iberian Conference on Pattern Recognition and Image Analysis**. [S.l.], 2013. p. 277–285.

GASPAR, J. A. d. C. P. **Visão para Robótica Móvel: Detecção de Obstáculos sobre Pavimento Plano**. 104 p. Tese (Doutorado) — UNIVERSIDADE TÉCNICA DE LISBOA, 1994.

GLEASON, J. Brisk (presented by josh gleason). In: **International Conference on Computer Vision**. [S.l.: s.n.], 2011.

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 kinect-style depth cameras for dense 3d modeling of indoor environments. **The International Journal of Robotics Research**, SAGE Publications, v. 31, n. 5, p. 647–663, 2012.

HIRZINGER, G. et al. The dlr-kuka success story: robotics research improves industrial robots. **IEEE robotics & automation magazine**, IEEE, v. 12, n. 3, p. 16–23, 2005.

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: **International Symposium on Robotics Research (ISRR)**. [S.l.: s.n.], 2011. v. 2.

ISACK, H.; BOYKOV, Y. Energy-based geometric multi-model fitting. **International journal of computer vision**, Springer, v. 97, n. 2, p. 123–147, 2012.

ISARD, M.; BLAKE, A. Condensation?conditional density propagation for visual tracking. **International journal of computer vision**, Springer, v. 29, n. 1, p. 5–28, 1998.

IZADI, S. et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: ACM. **Proceedings of the 24th annual ACM symposium on User interface software and technology**. [S.l.], 2011. p. 559–568.

JOLLIFFE, I.; JOLLIFFE, I. Principal Component Analysis. In: **Wiley StatsRef: Statistics Reference Online**. Chichester, UK: John Wiley & Sons, Ltd, 2014. Disponível em: <http://doi.wiley.com/10.1002/9781118445112.stat06472>.

KABUKA, M.; ARENAS, A. Position verification of a mobile robot using standard pattern. **IEEE Journal on Robotics and Automation**, IEEE, v. 3, n. 6, p. 505–516, 1987.

KAMILA, N. K. **Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing**. [S.l.]: Information Science Reference - Imprint of: IGI Publishing, 2015. ISBN 1466686545, 9781466686540.

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–506.

KELLY, A. **Mobile Robotics: Mathematics, Models, and Methods**. [S.l.]: Cambridge University Press, 2013.

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.

KLASING, K. et al. Comparison of surface normal estimation methods for range sensing applications. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**. [S.l.], 2009. p. 3206–3211.

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.

KRETZSCHMAR, H.; STACHNISS, C. Information-theoretic compression of pose graphs for laser-based slam. **The International Journal of Robotics Research**, SAGE Publications, v. 31, n. 11, p. 1219–1230, 2012.

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: **2009 IEEE 12th International Conference on Computer Vision**. IEEE, 2009. p. 2130–2137. ISBN 978-1-4244-4420-5. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459466>.

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.

LENARCIC, J.; BAJD, T.; STANIŠIĆ, M. M. **Robot mechanisms**. [S.l.]: Springer Science & Business Media, 2012.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 376–382, 1991. ISSN 1042296X. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=88147>.

LEUTENEGGER, S.; CHLI, M.; SIEGWART, R. Y. BRISK: Binary Robust invariant scalable keypoints. In: **2011 International Conference on Computer Vision**. IEEE, 2011. p. 2548–2555. ISBN 978-1-4577-1102-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126542>.

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. **2012 IEEE International Conference on Mechatronics and Automation**. [S.l.], 2012. p. 1921–1927.

LIU, Y. et al. Using em to learn 3d models of indoor environments with mobile robots. In: **ICML**. [S.l.: s.n.], 2001. v. 1, p. 329–336.

LOWE, D. Object recognition from local scale-invariant features. In: **Proceedings of the Seventh IEEE International Conference on Computer Vision**. IEEE, 1999. v. 2, p. 1150–1157 vol.2. ISBN 0-7695-0164-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410>.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International journal of computer vision**, Springer, v. 60, n. 2, p. 91–110, 2004.

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.

MARTINEZ, A.; FERNÁNDEZ, E. **Learning ROS for robotics programming**. [S.l.]: Packt Publishing Ltd, 2013.

MEIKLE, S.; YATES, R. Computer vision algorithms for autonomous mobile robot map building and path planning. In: **Proceedings of the Thirty-First Hawaii International Conference on System Sciences**. IEEE Comput. Soc, 1998. v. 3, p. 292–301. ISBN 0-8186-8255-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=656291>.

MEN, H.; GEBRE, B.; POCHIRAJU, K. Color point cloud registration with 4d icp algorithm. In: IEEE. **Robotics and Automation (ICRA), 2011 IEEE International Conference on**. [S.l.], 2011. p. 1511–1516.

MIKOLAJCZYK, K. et al. A performance evaluation of local descriptors. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 27, n. 10, p. 1615–1630, 2005. ISSN 0162-8828. Disponível em: <http://doi.ieeecomputersociety.org/10.110910.1109/TPAMI.2005.188>.

MIKSIK, O.; MIKOLAJCZYK, K. Evaluation of Local Detectors and Descriptors for Fast Feature Matching. **Pattern Recognition (ICPR), 2012 21st International Conference on**, IEEE,], n. Icpr, p. 2681–2684, 2012. ISSN 1051-4651.

MITRA, N. J. et al. Registration of point cloud data from a geometric optimization perspective. In: ACM. **Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing**. [S.l.], 2004. p. 22–31.

MONTGOMERY, D. C.; RUNGER, G. C.; CALADO, V. **Estatística aplicada e probabilidade para engenheiros**. [S.l.]: Grupo Gen-LTC, 2000.

MORENO, L. et al. A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors. **Journal of Intelligent and Robotic Systems**, Kluwer Academic Publishers, v. 34, n. 2, p. 135–154, 2002. ISSN 09210296. Disponível em: <http://link.springer.com/10.1023/A:1015664517164>.

NASEER, T. et al. Robust Visual Robot Localization Across Seasons Using Network Flows. **Aaai**, 2014. Disponível em: <http://ais.informatik.uni-freiburg.de/publications/papers/naseer14aaai.pdf>.

NEWCOMBE, R. A. et al. Kinectfusion: Real-time dense surface mapping and tracking. In: IEEE. **Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on**. [S.l.], 2011. p. 127–136.

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.

OLIVA, A.; TORRALBA, A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. **International Journal of Computer Vision**, Kluwer Academic Publishers, v. 42, n. 3, p. 145–175, 2001. ISSN 09205691. Disponível em: <http://link.springer.com/10.1023/A:1011139631724>.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5.

RABBANI, T.; HEUVEL, F. van den. Automatic point cloud registration using constrained search for corresponding objects. In: **Proceedings of 7th Conference on Optical**. [S.l.: s.n.], 2005. p. 3–5.

REINA, G. et al. Adaptive Kalman Filtering for GPS-based Mobile Robot Localization. In: **2007 IEEE International Workshop on Safety, Security and Rescue Robotics**. IEEE, 2007. p. 1–6. ISBN 978-1-4244-1568-7. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4381270>.

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: **2011 International Conference on Computer Vision**. IEEE, 2011. p. 2564–2571. ISBN 978-1-4577-1102-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126544>.

RUHNKE, M. et al. Compact rgbd surface models based on sparse coding. In: **AAAI**. [S.l.: s.n.], 2013.

RUSINKIEWICZ, S.; LEVOY, M. Efficient variants of the icp algorithm. In: IEEE. **3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on**. [S.l.], 2001. p. 145–152.

RUSU, R. B. Semantic 3d object maps for everyday manipulation in human living environments. **KI-Künstliche Intelligenz**, Springer, v. 24, n. 4, p. 345–348, 2010.

RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast point feature histograms (fpfh) for 3d registration. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**. [S.l.], 2009. p. 3212–3217.

RUSU, R. B. et al. Aligning point cloud views using persistent feature histograms. In: IEEE. **2008 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [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 implementaçao de algoritmos para localizaçao e mapeamento de robôs móveis baseada em computaçao reconfigurável**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computaçao, 2005.

SANCHES, V. L. M. **Um sistema de localização robótica para ambientes internos baseado em redes neurais.** 107 p. Tese (Doutorado) — Universidade de São Paulo, 2009.

SANDE, K. E. A. van de; GEVERS, T.; SNOEK, C. G. M. Evaluating Color Descriptors for Object and Scene Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 32, n. 9, p. 1582–1596, sep 2010. ISSN 0162-8828. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5204091>.

SARKER, B. R.; ISLAM, K. M. S. Relative performances of similarity and dissimilarity measures. **Computers & industrial engineering**, Elsevier, v. 37, n. 4, p. 769–807, 1999.

SAURER, O.; FRAUNDORFER, F.; POLLEFEYS, M. Visual localization using global visual features and vanishing points. **CEUR Workshop Proceedings**, v. 1176, p. 1–9, 2010. ISSN 16130073.

SCHAFFALITZKY, F.; ZISSERMAN, 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.

SE, S.; LOWE, D.; LITTLE, J. Local and global localization for mobile robots using visual landmarks. **Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on**, v. 1, p. 414–420 vol.1, 2001.

SE, S.; LOWE, D.; LITTLE, J. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. **The international Journal of robotics Research**, SAGE Publications Sage UK: London, England, v. 21, n. 8, p. 735–758, 2002.

SEHGAL, A.; CERNEA, D.; MAKAVEEVA, M. Real-time scale invariant 3d range point cloud registration. In: SPRINGER. **International Conference Image Analysis and Recognition**. [S.l.], 2010. p. 220–229.

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.

SICILIANO, B.; KHATIB, O. **Springer handbook of robotics**. [S.l.]: Springer Science & Business Media, 2008.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. Autonomous mobile robots. **Massachusetts Institute of Technology**, 2004.

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.

SMITH, M. et al. The new college vision and laser data set. **The International Journal of Robotics Research**, SAGE Publications, v. 28, n. 5, p. 595–599, 2009.

SPRICKERHOF, J. et al. An explicit loop closing technique for 6d slam. In: **ECMR**. [S.l.: s.n.], 2009. p. 229–234.

STURM, J. et al. A benchmark for the evaluation of RGB-D SLAM systems. In: **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE, 2012. p. 573–580. ISBN 978-1-4673-1736-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6385773>.

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.

SURMANN, H.; NÜCHTER, A.; HERTZBERG, J. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. **Robotics and Autonomous Systems**, v. 45, n. 3, p. 181–198, 2003. ISSN 09218890.

SURMANN, H.; NÜCHTER, A.; HERTZBERG, J. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. **Robotics and Autonomous Systems**, Elsevier, v. 45, n. 3, p. 181–198, 2003.

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**. Tese (Doutorado) — ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2001.

TRIGGS, B. et al. Bundle adjustment?a modern synthesis. In: SPRINGER. **International workshop on vision algorithms**. [S.l.], 1999. p. 298–372.

TUYTELAARS, T.; MIKOLAJCZYK, K. Local Invariant Feature Detectors: A Survey. **Foundations and Trends® in Computer Graphics and Vision**, Now Publishers Inc., v. 3, n. 3, p. 177–280, 2007. ISSN 1572-2740. Disponível em: <http://www.nowpublishers.com/article/Details/CGV-017>.

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.

WHELAN, T. et al. Elasticfusion: Real-time dense slam and light source estimation. **The International Journal of Robotics Research**, SAGE Publications, p. 0278364916669237, 2016.

WITTEN, I. H. I. H.; FRANK, E.; HALL, M. A. M. A. **Data mining : practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2011. 629 p. ISBN 9780123748560.

YANG, J.; LI, H.; JIA, Y. Go-icp: Solving 3d registration efficiently and globally optimally. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2013. p. 1457–1464.

YI, J. et al. IMU-based localization and slip estimation for skid-steered mobile robots. In: **2007 IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE, 2007. p. 2845–2850. ISBN 978-1-4244-0911-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4399477>.

ZHANG, L.; GHOSH, B. K. Line segment based map building and localization using 2D laser rangefinder. **Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on**, IEEE, v. 3, p. 2538–2543 vol.3, 2000.

ZHANG, L. et al. Real-time human motion tracking using multiple depth cameras. In: IEEE. **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2012. p. 2389–2395.

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.

ZHANG, Z. Microsoft kinect sensor and its effect. **IEEE multimedia**, IEEE, v. 19, n. 2, p. 4–10, 2012.

Zhenhua Wang, Z.; FAN, B.; WU, F. Local Intensity Order Pattern for feature description. In: **2011 International Conference on Computer Vision**. IEEE, 2011. p. 603–610. ISBN 978-1-4577-1102-2. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126294>.

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.