UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MIRELLA SANTOS PESSOA DE MELO

MAPEAMENTO DE REGIÃO NAVEGÁVEL A PARTIR DE UM SISTEMA SLAM E SEGMENTAÇÃO DE IMAGEM

Recife
2021

MIRELLA SANTOS PESSOA DE MELO

MAPEAMENTO DE REGIÃO NAVEGÁVEL A PARTIR DE UM SISTEMA SLAM E SEGMENTAÇÃO DE IMAGEM

Este trabalho foi apresentado à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia da Computação

Orientadora: Edna Natividade da Silva Barros

Recife

2021

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M528m Melo, Mirella Santos Pessoa de
Mapeamento de região navegável a partir de um sistema SLAM e segmentação de imagem / Mirella Santos Pessoa de Melo. – 2021.
122 f.: il., fig., tab.

Orientadora: Edna Natividade da Silva Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências.

1. Engenharia da computação. 2. Navegação. I. Barros, Edna Natividade da Silva (orientadora). II. Título.

621.39 CDD (23. ed.) UFPE - CCEN 2021 – 171

MIRELLA SANTOS PESSOA DE MELO

"**MAPEAMENTO DE REGIÃO NAVEGÁVEL A PARTIR DE UM SISTEMA SLAM E SEGMENTAÇÃO DE IMAGEM**"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 13/08/2021.

Orientadora: Edna Natividade da Silva Barros

BANCA EXAMINADORA

Prof. Dra. Edna Natividade da Silva Barros
Centro de Informática / UFPE

Prof. Dr. Hansenclever de Franca Bassani
Centro de Informática / UFPE

Prof. Dr. Francisco Paulo Magalhães Simões
Departamento de Computação / UFRPE

Dedico esta dissertação a todos àqueles que acreditam na educação e no seu poder de transformar vidas.

AGRADECIMENTOS

Em determinado momento do meu mestrado, acreditei veementemente que o tão sonhado momento de entregar esta dissertação não seria possível. Mas, Deus colocou no meu caminho pessoas certas e na hora certa. Uma delas foi a professora Edna, que me acolheu num momento delicado e me orientou de forma muito séria e comprometida. Portanto, eu agradeço primeiramente a Deus e segundamente a ela pela confiança e pela oportunidade dada a mim. Agradeço os aprendizados e a troca semanal, o ouvido atento e curioso. Minhas sextas foram melhores com os nossos encontros. Obrigada.

Agradeço aos meus pais, Rogério e Marquilene Pessoa de Melo, que nunca mediram esforços para dar a mim e a meus irmãos a melhor educação possível. É através dela que chego aqui hoje, e realizo mais uma etapa do meu sonho. Meus pais, obrigada por sempre acreditarem em mim e apoiarem as minhas escolhas. Agradeço aos meus avós Raimundo e Artimisse Santos, que me inspiram a ser uma pessoa melhor, a ver o próximo com empatia e respeito. O amor e a parceria de vocês é minha maior referência de vida. Agradeço ao meu marido André Ferraz que nunca deixou de acreditar em mim nos momentos em que eu mesma duvidei. Que me motivou e acolheu as minhas mais expostas fraquezas. Que me inspirou a ser melhor, a ter calma e continuar. Agradeço à família Ferraz, que acompanhou minha trajetória, que se preocuparam e vibraram comigo em cada altos e baixos. Agradeço em especial ao meu sogro Carlos Ferraz que por muitas vezes atuou como professor que é, e me orientou nessa caminhada.

Agradeço aos amigos Caio Brito e Pammela Araújo pelo contato amigo e pelos receios ouvidos e amenizados. Agradeço ao pesquisador Lucas Cambuim por toda a parceria e troca durante esse período e, também, ao amigo Airton Sampaio pelos inúmeros auxílios técnicos. Agradeço ao laboratório Voxar pela primeira porta aberta na computação, de onde conheci o tema dessa dissertação. Agradeço a todos, sem exceção, que de alguma forma me ajudaram a estar fechando esse ciclo. Eu jamais chegaria aqui sozinha. Muito obrigada!

RESUMO

Agentes robóticos que se utilizam de algoritmos de localização e mapeamento simultâneos (SLAM) realizam a construção incremental do mapa de um ambiente desconhecido enquanto, simultaneamente, determinam sua localização dentro desse mapa. Estes são chamados de sistemas SLAM visual (vSLAM) quando se utilizam de dados proveniente de uma câmera e, pela forma com que as imagens são interpretadas, são categorizados em método direto ou indireto. O vSLAM de método indireto é eficiente, rápido e pode oferecer um sistema de localização preciso; por outro lado, representa o ambiente mapeado através de uma nuvem de pontos esparsa, sendo esta imprópria para o planejamento de rotas. Portanto, nosso trabalho teve como objetivo desenvolver um módulo capaz de associar o mapeamento esparso do vSLAM indireto com o planejamento de rotas para navegação, tendo como prioridade o baixo custo computacional, fazendo uso apenas de um sensor de câmera estéreo. A representação adotada para modelar o ambiente é um mapa de grade de ocupação 2D - OGM 2D, uma das opções predominantemente utilizadas na robótica. Para criar o OGM, associamos o estágio de mapeamento do vSLAM com uma técnica de segmentação de chão. A proposta representa um módulo complementar que além de transformar a nuvem de pontos esparsa em um OGM 2D, também resulta numa nuvem de pontos segmentada entre chão e não-chão. Avaliações sobre o mapa gerado foram feitas em ambientes sintéticos e reais, considerando algoritmos de planejamento de rota, sobreposição de mapas e métricas computacionais. Resultados revelam mapas com alta precisão enquanto exigem baixíssimo acréscimo do consumo de memória, e tempo de processamento que permite que a aplicação seja executada junto ao SLAM em tempo real. O mapa gerado permite um elo entre um algoritmo SLAM de mapeamento esparso e atividades de navegação.

Palavras-chaves: sistema SLAM; navegação; mapa de ocupação.

ABSTRACT

Robotic agents that use simultaneous localization and mapping algorithms (SLAM) perform the incremental construction of the map of an unknown environment while simultaneously determining its location within that map. Visual SLAM systems (vSLAM) use data from a camera and can be categorized as a direct or indirect method based on how it interprets the images. Indirect method vSLAM is efficient, fast, and can provide an accurate localization system; on the other hand, it represents the environment mapped through a sparse point cloud, which is unsuitable for route planning. Therefore, our work aimed to develop a module capable of associating the sparse mapping of indirect vSLAM with route planning for navigation, prioritizing the low computational cost, and using only a stereo camera sensor. The representation adopted to model the environment is a 2D occupation grid map - 2D OGM, one of the options predominantly used in robotics. To create the OGM, we associated the vSLAM mapping stage with a ground segmentation technique. Thus, the proposal represents a complementary module that, in addition to transforming the sparse point cloud into a 2D OGM, also results in a point cloud segmented between ground and non-ground. Evaluations of the generated map were made in synthetic and real environments, considering route planning algorithms, map overlays, and computational metrics. Results reveal maps with high accuracy while requiring a low memory consumption and processing time, allowing the application to run alongside SLAM in real-time. Furthermore, the generated map enables a link between a sparse mapping SLAM algorithm and navigation activities.

Keywords: SLAM; navigation; occupancy grid map.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BA	Bundle Adjustment
BoW	Bags of Visual Words
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HVS	Human Vision System
OGM	Occupancy Grid Map
ORB	Oriented FAST and rotated BRIEF
PCL	Point Cloud Library
PRM	Probabilistic Road Map
PTAM	Parallel Tracking and Mapping
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
ROS	Robot Operating System
SAD	Sum of Absolute Differences
SDF	Signed Distance Function
SGM	Semi Global Matching
SLAM	Simultaneous Localization and Mapping
TSDF	Truncated Signed Distance Function
vSLAM	Visual Simultaneous Localization and Mapping

LISTA DE SÍMBOLOS

α	Letra grega minúscula Aplha
β	Letra grega Beta
θ	Letra grega Theta
$\geq$	Maior ou Igual
$\leq$	Menor ou Igual
π	Letra Grega Pi
ϕ	Letra grega Phi
Ω	Letra grega Ômega

SUMÁRIO

1 INTRODUÇÃO

Agentes robóticos que se utilizam de algoritmos de localização e mapeamento simultâneos - SLAM, realizam a construção incremental do mapa de um ambiente desconhecido enquanto, simultaneamente, determinam sua localização dentro desse mapa. Sistemas SLAM representam uma poderosa ferramenta para autonomia robótica, e mais expressivamente na última década, os esforços da comunidade se converteram em resultados e avanços promissores (SAPUTRA; MARKHAM; TRIGONI, 2018; LI; WANG; GU, 2018). No entanto, ainda há muito campo em aberto e desafios para serem resolvidos, onde o sistema ideal contempla a associação entre a estimativa de uma localização robusta, a representação densa do ambiente, e a compreensão semântica do mesmo (CADENA et al., 2016; AZZAM et al., 2020; XIA et al., 2020).

No caso de um algoritmo SLAM visual - vSLAM, são utilizados dados capturados de uma câmera. Para ser considerado um sistema *completo*, é preciso ser (a) capaz de estimar a posição da câmera por meio de imagens juntamente com (b) a capacidade de manter consistência do mapa local, e ao (c) reconhecer uma revisita, (d) de forma global, otimizar mapa e trajetória previamente estimados. Pela forma com que as imagens são utilizadas, o sistema é categorizado em método *direto* ou *indireto*. Algumas considerações de como cada uma dessas abordagens se comporta com relação ao mapeamento, à localização e ao custo computacional, bem como suas respectivas vantagens e desvantagens, são discutidas a seguir.

Os primeiros trabalhos de método indireto, isto é, baseado em *características* (DAVISON et al., 2007; KLEIN; MURRAY, 2007; MUR-ARTAL; MONTIEL; TARDOS, 2015) ganharam grande adesão da comunidade por serem precisos em termos de localização; por apresentarem certa robustez mesmo em ambientes dinâmicos, seja em termos de agentes móveis na cena, e.g pessoas, ou variação de iluminação; e por sua funcionalidade em tempo real mesmo com uso de Central Processing Unit (CPU). Características (*features*, em inglês) são pontos específicos no quadro de imagem que exibem propriedades bem definidas, exclusivas e de fácil reconhecimento. Sua extração é feita de forma esparsa na imagem, ou seja, apenas para regiões que se enquadram nas exigências. Dado que o sistema SLAM acumula erro nas estimativas da trajetória e do mapeamento, o uso de *features* é um grande aliado para (a) o reconhecimento de uma revisita e (b) para a correção do acúmulo de erro, ambos em tempo real. Essa vantagem se estende, inclusive, para ambientes de larga escala (FILLIAT, 2007).

O mapa gerado das primeiras abordagens, que é esparso, garante um processamento rápido em todos os sentidos, ao mesmo tempo que representa uma desvantagem para atividades em que é necessária uma maior percepção do ambiente. Portanto, há trabalhos que tomam como base ou se inspiram nas propostas de método indireto para enriquecer o mapeamento, chegando numa representação densa (LABBÉ; MICHAUD, 2019; LI et al., 2018) ou semi-densa do ambiente, exemplificado pelo trabalhos (WANG et al., 2018; MUR-ARTAL; TARDÓS, 2015; WANG et al., 2017) e pelo ORB-SLAM3 (CAMPOS et al., 2021), o qual adiciona ao sistema um sensor

inercial.

Em contra partida, a maioria das propostas de métodos direto não sofrem desse problema, isto porque conseguem reconstruir o ambiente de forma densa (estimativa de profundidade para todos os *pixels* da imagem) (WHELAN et al., 2015b; WHELAN et al., 2016; WHELAN et al., 2015a), ou semi-densa (estimativa de profundidade apenas para regiões de alto gradiente) (ENGEL; SCHÖPS; CREMERS, 2014), à medida que estimam a odometria com robustez. Pela forma com que mapeiam, isto é, levando em consideração a intensidade ou gradiente dos *pixels*, o sistema se torna mais susceptível a erros em cenas dinâmicas, seja com relação à variação na iluminação ou objetos móveis na cena. Sistemas de método direto são grandes aliados quando o intuito é a reconstrução 3D, especialmente em uma cena controlada ou ainda, quando não há *features* suficientes para serem detectadas, isto é, em ambientes de baixa texturização.

Quando se trata de agentes robóticos, alguns problemas de processamento em tempo real podem aparecer. Propostas de reconstrução densa geralmente requerem uso de Graphics Processing Unit (GPU), onde testes são realizados em cenas de pequena escala, por exemplo, no tamanho de um cômodo (WHELAN et al., 2015b; WHELAN et al., 2016; WHELAN et al., 2015a). De forma geral, para propostas de mapeamento denso a otimização do mapa global ainda é um desafio presente em termos de custo computacional (DAI et al., 2017; TAKETOMI; UCHIYAMA; IKEDA, 2017) uma vez que toda a configuração geométrica é alterada. Um exemplo de tentativa de reduzir o custo desse método é a partir do mapeamento semi-denso (ENGEL; SCHÖPS; CREMERS, 2014) e até mesmo, esparso (GAO et al., 2018).

Até aqui, foram evidenciadas as diferenças entre os métodos vSLAM, as formas de mapeamento e seus respectivos impactos sobre custo computacional, bem como os pontos fracos e fortes. Para pontuar com relação à estimativa de localização desses sistemas, a Tabela 1 traz apenas alguns resultados extraídos (a) do trabalho descrito em (FILIPENKO; AFANASYEV, 2018), que considera em sua avaliação um ambiente *indoor* (base de dados própria, nomeada de *indoor* na Tabela 1), e (b) do trabalho ORB-SLAM3, que foca em ambientes *indoor* e *outdoor*. Ambos avaliam a raiz quadrada do erro-médio da trajetória (RMSE). Outros trabalhos como o RTAB-Map (LABBÉ; MICHAUD, 2019), também trazem uma comparação extensa, a qual avalia o erro absoluto da trajetória (ATE).

Tanto pela Tabela 1, quanto por uma análise minuciosa dos trabalhos, fica evidenciado que a precisão na localização dos sistemas de método direto e indireto são bastante competitivas, sendo difícil de definir qual deles é o melhor nesse aspecto, principalmente por existir uma infinidade de cenas possíveis, sejam elas *indoor* ou *outdoor*.

Tabela 1 – Comparativo entre sistemas SLAM de método direto e indireto considerando diferentes modelos de câmera e cenários. A avaliação leva em conta a raiz quadrada do erro-médio (RMSE) entre a trajetória estimada e o *ground truth*.

SLAM	Método	Sensor	Base de dados	RMSE [m]
ORB-SLAM	Indireto	Monocular	Indoor	**0.166**
LSD-SLAM	Direto	Monocular	Indoor	0.301
ORB-SLAM2	Indireto	Estéreo	Indoor	0.190
RTAB-Map	Indireto	Estéreo	Indoor	**0.163**
ORB-SLAM2	Indireto	Estéreo	EuRoC (V1_easy)	**0.035**
LSD-SLAM	Direto	Estéreo	EuRoC (V1_easy)	0.066
ORB-SLAM2	Indireto	Estéreo	EuRoC (V1_difficult)	**0.048**
LSD-SLAM	Direto	Estéreo	EuRoC (V1_difficult)	0.089
ORB-SLAM2	Indireto	RGB-D	TUM (fr1/desk)	**0.016**
Elastic-Fusion	Direto	RGB-D	TUM (fr1/desk)	0.020
ORB-SLAM2	Indireto	RGB-D	TUM (fr3/office)	**0.010**
Elastic-fusion	Direto	RGB-D	TUM (fr3/office)	0.017

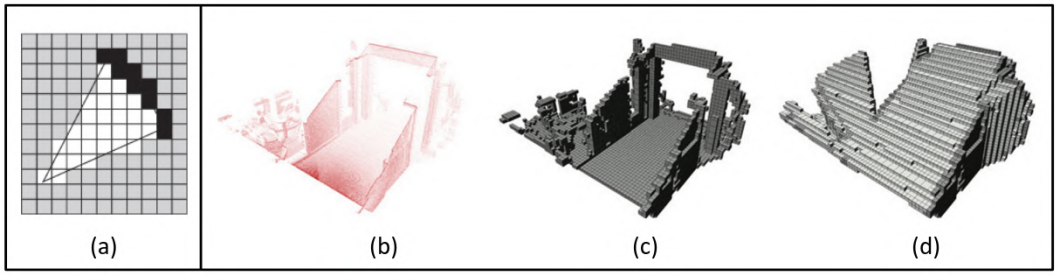
Fonte: autora.

1.1 MAPEAMENTO E NAVEGAÇÃO

No que diz respeito à autonomia robótica, especialmente quando há limitações de hardware e a aplicação é voltada para ambientes de larga escala, sistemas vSLAM de método indireto com mapeamento esparso representam uma alternativa promissora. Contudo, a reprodução esparsa da cena resulta na falta de elo entre esses sistemas e uma representação de mapa direcionado para a atividade de navegação, como planejamento de rota e desvio de obstáculos (LING; SHEN, 2017). Mas, como representar um mapa do ambiente durante o mapeamento de tal forma que favoreça a navegação e, consequentemente, o alcance da autonomia de navegação?

A primeira proposta, apresentada por Alberto Elfes, é a representação 2D do mapa em grades de ocupação (em inglês, *occupancy grid map* - OGM) com grades de tamanho fixo (ELFES, 1989; THRUN, 2003). Mapas binários apresentam para cada grade a informação de "ocupado", "livre" ou "não definido" (por exemplo, quando o sensor não cobriu a região), como ilustrado na Figura 1.a. No caso do OGM de probabilidades, cada grade possui a probabilidade do estado. OGM está entre as abordagens mais usadas para navegação, exploração, bem como para a fusão de sensores no domínio da robótica (KHAN, 2017). Existem diversos algoritmos criados sobre essa representação para realizar atividades de navegação (LAVALLE, 2006; MACENSKI et al., 2020).

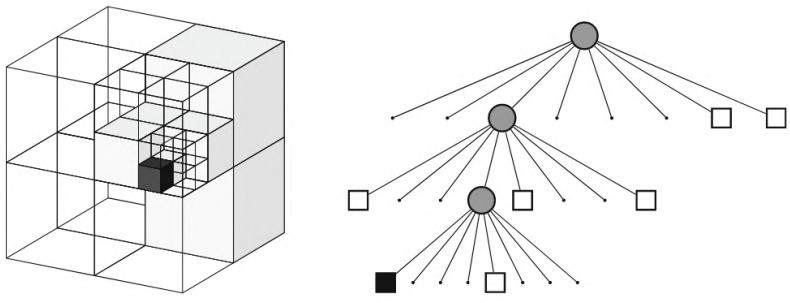
Figura 1 – Representação visual de (a) OGM 2D, (b) nuvem de pontos proveniente de um sensor de laser, (c) mapa de *voxels* ocupados, (d) mapa de *voxels* livres.



Fonte: (HORNUNG et al., 2013)

A Figura 1 ilustra (b) uma nuvem de pontos proveniente de um sensor de laser, (c) o mapa de *voxels* ocupados e (d) o mapa de *voxels* livres. *Voxel* é o equivalente à grade 2D, mas voltado para modelagem no espaço tridimensional, também de tamanho fixo (ROTH-TABAK; JAIN, 1989). No entanto, trazer grades de ocupação 2D para um cenário 3D leva a um grande custo computacional. Com isso, a representação em octomap (MEAGHER, 1982; HORNUNG et al., 2013) foi amplamente adotada. Esta abordagem se baseada em árvores de probabilidades de ocupação, como mostra a Figura 2, onde o modelo volumétrico é subdividido recursivamente em oito sub-volumes até que um determinado tamanho mínimo de *voxel* seja alcançado, o qual determina a resolução da árvore. Essa estrutura permite que grandes blocos do espaço com a mesma probabilidade sejam representados por um único *voxel* de tamanho maior, diminuindo, assim, a quantidade de memória necessária para representar áreas desconhecidas ou livres (geralmente, a maioria).

Figura 2 – Modelo volumétrico à esquerda e a representação em árvore correspondente à direita.



Fonte: (HORNUNG et al., 2013)

É importante citar que, além das representações volumétricas citadas, há também outras alternativas, exemplificadas por mapas de elevação (KWEON et al., 1989), mapas de superfície de vários níveis (TRIEBEL; PFAFF; BURGARD, 2006), mapas topológicos e nuvem de pontos (utilizada pelos sistemas SLAM mencionados).

1.2 PROPOSTA

Os trabalhos de vSLAM discutidos representam o ambiente mapeado através de uma nuvem de pontos, sendo esta imprópria para o planejamento de rotas (em inglês, *motion planning*). Portanto, este trabalho tem como objetivo desenvolver um mecanismo que estabeleça um elo entre um sistema vSLAM e atividades de navegação, tendo como foco a redução do custo computacional. Com isso, sistemas embarcados em agentes robóticos, que apresentam limitações de hardware, são favorecidos. Para corroborar com nosso foco, a proposta exige apenas um sensor de câmera estéreo, sendo uma solução de baixo custo e de baixo consumo de energia.

Dentre os métodos apresentados de vSLAM, os de método indireto com mapeamento esparso são computacionalmente mais baratos e, por consequência, mais rápidos à medida que exibem excelente robustez em termos de localização. No entanto, entendemos que o mapa esparso produzido representa uma desvantagem para a autonomia robótica, mais especificamente no que diz respeito à navegação. Portanto, nosso intuito é alcançar uma maior densidade de mapa de forma que este seja apropriado para algoritmos de planejamento de rotas. A representação adotada para modelar o ambiente é um mapa de grade de ocupação 2D - OGM 2D (ELFES, 1989; THRUN, 2003), uma das opções predominantemente utilizadas em robótica (KHAN, 2017). Vale salientar que a proposta é direcionada para robôs móveis de chão, cuja bidimensionalidade é suficiente.

O método proposto, denominado de Segfloor, tem as seguintes finalidades:

- gerar o OGM a partir da associação do estágio de mapeamento do vSLAM com uma técnica de segmentação de imagem;
- representar um módulo complementar que transforma a nuvem de pontos esparsa em um OGM 2D e também numa nuvem segmentada entre *chão* e *não-chão*;
- ser um módulo que leva à considerável redução do uso de memória mesmo para uma cena de larga escala, devido ao uso exclusivo e direto da nuvem esparsa, isto é, sem a necessidade de antes gerar uma representação de pontos densa, como feito no trabalho (LABBÉ; MICHAUD, 2019) (usado para fins de comparação com o nosso sistema).

Para a avaliar a técnica, coletamos dados em ambiente real e sintético contendo as imagens Red, Green, Blue (RGB) sincronizadas da câmera estéreo, o *ground truth* do mapa grade de ocupação (quando possível) e a respectiva planta baixa. Para a cena sintética, as informações de odometria também foram coletadas.

A proposta apresentada tem como pré-requisitos os mesmos do sistema SLAM baseado em *features*, isto é, que o ambiente a ser mapeado apresente regiões com textura para que, assim, seja possível a extração de *features* e subsequente mapeamento e localização. Por exemplo, uma cena somente com paredes e chão totalmente brancos, sem áreas "marcantes",

como tomadas, manchas, rachaduras, relevos e etc, indica uma limitação. Um requerimento adicional consiste na necessidade de textura (ainda que baixa) especificamente na região navegável.

A dissertação está estruturada da seguinte forma. Nos Capítulos 2 e 3 de conceitos básicos, apresentamos os principais conteúdos teóricos envolvidos neste trabalho, que vão desde o detalhamento técnico do sensor utilizado (câmera estéreo) até o sistema SLAM adotado. No Capítulo 4, abordamos os trabalhos relacionados, os quais também geram mapas adequados para navegação a partir de um sistema vSLAM. No Capítulo 5, discutimos sobre a etapa de desenvolvimento da técnica proposta. No Capítulo 6, detalhamos qual a metodologia e quais as métricas utilizadas para avaliar nosso método. No Capítulo 7, apresentamos e discutimos os resultados. Por fim, no Capítulo 8, trazemos a conclusão e intenções de trabalhos futuros.

2 IMAGEM E PROFUNDIDADE

O conceito que deu início à fotografia e ao processo de captura e aquisição de uma imagem se deu no século IX com a *câmera obscura*, fisicamente representada por uma sala toda fechada com um pequeno orifício e uma lente responsável por convergir os raios de luz, projetando-os na parede oposta à abertura. Apenas na primeira metade do século XIX, com a descoberta de materiais sensíveis à luz, a captura da imagem propriamente dita se tornou possível. Desde então, a câmera evoluiu de uma caixa simples que tirava fotos borradas para os minicomputadores de alta tecnologia, alta resolução e fidelidade de cores.

No entanto, na aquisição de uma imagem bidimensional, dado uma cena tridimensional, ocorre a perda de uma dimensão: a profundidade. A recuperação desse dado através de um conjunto de imagens e de correspondência de pontos ainda é um campo em aberto na ciência (SZELISKI, 2010). Diversas linhas de pesquisas dependem dessa estimativa, como as utilizadas neste trabalho, relativas ao sistema de localização e mapeamento simultâneo (SLAM) e *Structure from Motion*[1], as quais tomam proveito dos estudos da geometria projetiva (MUNDY; ZISSERMAN, 1992).

Para recuperar a informação espacial, o presente trabalho faz uso de um sistema estereoscópico de duas câmeras. Dado que esse arranjo é inspirado na natureza, a Seção 2.1 traz uma breve descrição do sistema de visão humano associado à interpretação de profundidade. Em seguida, a Seção 2.2 aborda como o sistema de câmera estéreo recupera o dado 3D. Para isso, duas macro atividades se fazem necessárias: (a) a definição dos parâmetros do sistema, abordado na Seção 2.3; (b) o problema de correspondência estéreo, discutido na Seção 2.4. Por fim, a Seção 2.5 trata sobre um dos tipos de correspondência estéreo, a densa, e a Seção 2.6 descreve a técnica de correspondência estéreo adotada neste trabalho. Vale salientar que ambas atividades (a) e (b) trazem conceitos fundamentais para o entendimento do Capítulo 3, que diz respeito ao sistema SLAM.

2.1 SISTEMA DE VISÃO HUMANA

Um sistema capaz de capturar duas imagens a partir de diferentes pontos de vista, pode usufruir de uma importante informação: profundidade. Esse é um artifício utilizado por nós humanos, e todos os seres vivos que se utilizam de um par (ou mais) de olhos apontados para frente. No sistema visual humano (em inglês, human visual system - Human Vision System (HVS)), ilustrado pela Figura 3 de forma sucinta, é possível observar os raios de luz atingindo a retina ocular, passando pelos pontos P_1, P_2, H e Q.

Esses raios de luz atravessam, entre outros elementos, o cristalino, o qual é responsável por focar corretamente os raios na retina, se ajustando para uma imagem mais longe ou mais

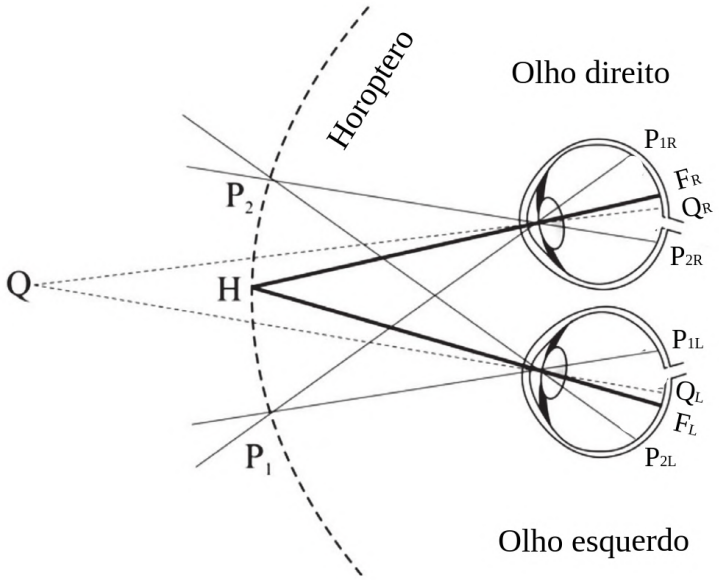
[1] Algoritmos de correspondência cuja finalidade é de extrair a estrutura 3D de uma cena.

perto. Esta é uma das diversas formas utilizadas por nós humanos para estimar a profundidade, conhecida como profundidade por foco ou desfoco (em inglês, *depth from focus or defocus*).

Ainda sobre o olho humano, tem-se a fóvea F, localizada na retina, que é uma região que se diferencia por apresentar maior nitidez dos raios que ali alcançam, ou seja, indica a visão central ou ponto de fixação. Isso é exemplificado na Figura 3, onde o ponto de luz H é interpretado na fóvea direita F_R e esquerda F_L com maior riqueza de informações.

Os raios incidentes na retina são divididos entre *correspondentes* e *não-correspondentes*. O primeiro caso ocorre quando a distância do ponto de luz incidente na retina direita até a fóvea direita é igual a distância desse mesmo ponto de luz incidente na retina esquerda até a fóvea esquerda. Isso é o que acontece, por exemplo, com os pontos P_1, P_2 e quaisquer pontos sob o horoptero, ou seja, $\overline{P_{1R}F_R} = \overline{P_{1L}F_L}$ e $\overline{P_{2R}F_R} = \overline{P_{2L}F_L}$. O mesmo não acontece com o ponto Q, que inclusive atinge a retina em direções opostas às fóveas. Portanto, demais pontos fora do horoptero representam raios não-correspondentes. A diferença relativa da distância da fóvea para cada um desses pontos é denominada *disparidade retiniana*. Para o caso ilustrado, têm-se que a disparidade entre $\overline{P_{1R}F_R}$ e $\overline{P_{1L}F_L}$ é igual a zero, enquanto que entre $\overline{Q_RF_R}$ e $\overline{Q_LF_L}$ é diferente de zero.

Figura 3 – Ilustração do conceito de disparidade retiniana considerando um par de olhos.



Adaptada: (CYGANEK; SIEBERT, 2011)

Uma forma mais intuitiva de entender como a disparidade retiniana funciona é colocando um objeto em frente aos olhos e fechando um deles em alternância. Dessa forma, é possível perceber que o objeto age como se houvesse movido mesmo parado. Esse deslocamento percebido é justamente a disparidade.

Sabe-se que essa disparidade retinal é um dos fatores levados em consideração pelo HVS para inferir a distância relativa dos elementos no mundo tridimensional. Embora ainda seja incerto como exatamente o cérebro realiza a sua detecção, pesquisas apontam a influência

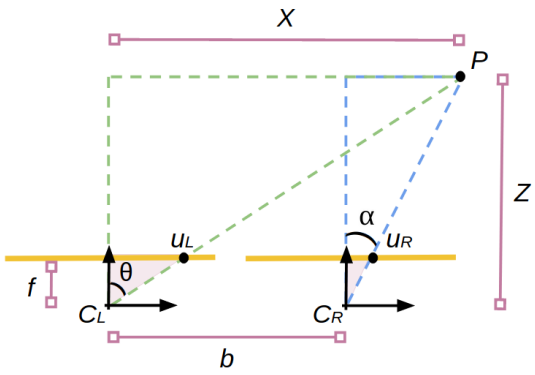
de texturas, formatos e variação da iluminação atuando num mecanismo de correspondências. Isso associado ao tamanho relativo dos objetos com outros na cena, bem como sua movimentação, perspectiva linear, a própria vivência do indivíduo, dentre outros fatores que, quando interpretados juntos, induzem à estimativa da profundidade (CYGANEK; SIEBERT, 2011).

2.2 SISTEMA ESTEREOSCÓPICO E A PROFUNDIDADE

Sistemas de hardware tentam replicar o HVS a partir de duas câmeras com posicionamento fixo entre si, compondo um sistema estereoscópico. A estimativa de profundidade é possível através do método conhecido por *triangulação*. A Figura 4, em conjunto com as Equações 2.1 à 2.6, ilustram uma das formas de interpretação desse processo por meio de duas imagens obtidas de diferentes pontos de vista.

Na Figura 4, o ponto $P(X, Z)$ está representado em função do sistema de coordenadas da câmera esquerda C_L; b representa a distância entre o sistema de coordenadas da câmera esquerda e direita C_R; u_L e u_R são os pontos de interseções do ponto P no plano da imagem esquerda e direita respectivamente; f é a distância de C_L (ou C_R) até sua projeção ortogonal no plano de imagem respectivo. O que exatamente cada um desses elementos significa será detalhados na Seção 2.3 de parâmetros da câmera e do sistema, mas por enquanto, abstraímos.

Figura 4 – Um ponto P no espaço projetado em dois planos de imagem. As variáveis destacadas fazem parte da contextualização do método de triangulação.



Fonte: autora

Dado os elementos destacados e os triângulos tracejados em verde e azul na Figura 4, é possível escrever as Equações 2.1 e 2.2 a partir das relações trigonométricas no triângulo retângulo.

$$\frac{X}{Z} = tan(\theta) \qquad (2.1) \qquad\qquad \frac{X-b}{Z} = tan(\alpha) \qquad (2.2)$$

Isolando X em ambas as Equações 2.1 e 2.2, e igualando esta variável, chega-se na Equação 2.3.

$$Z[tan(\theta) - tan(\alpha)] = b \tag{2.3}$$

Sabe-se que $tan(\theta)$ e $tan(\alpha)$ também podem ser representados em função da interseção do ponto P no plano da imagem (triângulos destacados em rosa claro na Figura 4), como mostrado nas Equações 2.4 e 2.5. Por conveniência, a origem do sistema de coordenadas do plano da imagem está localizada na projeção perpendicular de C_L (ou C_R) no plano (ilustrado tridimensionalmente na Figura 6). No entanto, sua origem é comumente empregada no canto superior esquerdo.

$$\frac{u_L}{f} = tan(\theta) \tag{2.4}$$

$$\frac{u_R}{f} = tan(\alpha) \tag{2.5}$$

Finalmente, substituindo na Equação 2.3 os respectivos valores de $tan(\theta)$ e $tan(\alpha)$ das Equações 2.4 e 2.5, chega-se na Equação 2.6.

$$Z = \frac{fb}{u_L - u_R} \tag{2.6}$$

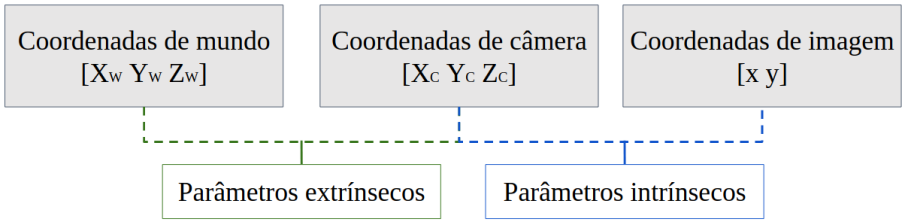
Portanto, fica evidente que o sucesso da estimativa da profundidade a partir da técnica de triangulação depende de dois fatores:

- os parâmetros do sistema e do modelo das câmeras - abordado na Seção 2.3;
- a correta associação dos pontos correspondentes nas duas imagens - apresentado na Seção 2.4.

2.3 PARÂMETROS DA CÂMERA E DO SISTEMA ESTEREOSCÓPICO

A modelagem, ou simplesmente, os parâmetros do sistema, podem ser divididos entre parâmetros intrínsecos, isto é, referente às informações internas do dispositivo, e parâmetros extrínsecos, que envolve notações relevantes relativas aos componentes externos à câmera. Como ilustrado na Figura 5, cada uma dessas classificações torna possível a conversão dos diferentes sistemas de coordenados envolvidos: coordenadas de mundo, de câmera e de imagem. Dado que essas transformações geométricas no âmbito da visão computacional são usualmente baseadas em coordenadas homogêneas, a Seção 2.3.1 traz uma breve recapitulação do assunto, facilitando a compreensão de seções subsequentes.

Figura 5 – Ilustração dos sistemas de coordenadas de mundo, de câmera e de imagem, e o respectivo conjunto de parâmetros responsável pela transformação de coordenadas entre esses sistemas.



Fonte: autora.

2.3.1 Sistema de Coordenadas Homogêneas

Dado um ponto $P(x, y)$ em $\mathbb{R}^2$ no espaço cartesiano, a sua respectiva representação em coordenadas homogêneas é acrescida de uma dimensão, logo, $\widetilde{P}(x, y, w)$ em $\mathbb{R}^3$, onde o elemento adicionado é chamado de peso, e o til (~) é empregado para indicar que o ponto em questão está representado em coordenadas homogêneas. Já a conversão de um ponto $\widetilde{P}(u, v, w)$ no sistema de coordenadas homogêneas para o cartesiano se dá a partir da divisão do peso w pelos demais elementos, seguido da diminuição de uma dimensão, neste caso, $P(u/w, v/w)$. É importante observar que cada linha em coordenadas homogêneas resulta em um único ponto no espaço cartesiano. Isto é possível de ser observado através de um ponto $\widetilde{P}(ku, kv, kw)$ com k assumindo qualquer valor real diferente de zero.

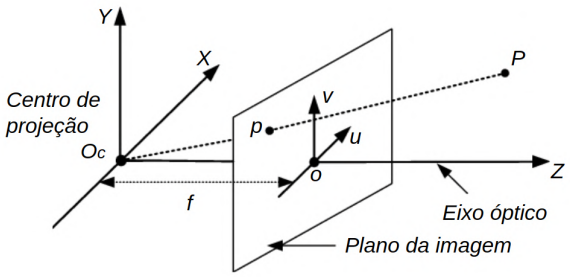
As vantagens de usar esse sistema consistem em (a) cálculos de transformação realizados a partir de operações entre matrizes e equações lineares; e (b) pela possibilidade de representar pontos no infinito, isto é, quando $w = 0$.

2.3.2 Parâmetros Intrínsecos

Começaremos tratando dos parâmetros intrínsecos, relativo à câmera em si. Os elementos que fazem parte da definição desse conjunto permitem a mudança do sistema de coordenada (ou *frame*) da câmera (3D) para o sistema de coordenada do plano da imagem (2D) e vice-versa.

O presente trabalho se utiliza de duas câmeras do modelo mais simples e amplamente utilizado, a *pinhole*. Por conveniência, esse modelo é usualmente representado com o plano de imagem entre o centro de projeção da câmera O_c (onde convergem os raios de luz) e a cena, como ilustrado na Figura 6. Uma segunda facilidade adotada é a do centro de coordenada do plano da imagem o coincidindo com a projeção ortogonal de Oc no plano.

Figura 6 – Ilustração do modelo de câmera pinhole, descrita com o plano da imagem entre o centro de projeção e a cena, preservando a orientação do ambiente.



Adaptada: (HARTLEY; ZISSERMAN, 2004).

Sendo assim, em coordenadas homogêneas, o ponto $\widetilde{P}(x, y, z, 1)$ com respeito ao *frame* da câmera, pode ser relacionado à $\widetilde{p}(u, v, 1)$ no *frame* da imagem pela Equação 2.7, onde f representa o comprimento focal da câmera e λ o fator de escala que permite o ponto p transitar por toda a linha $\overline{O_c P}$.

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

No entanto, a representação completa dos parâmetros intrínsecos deve considerar elementos adicionais, ilustrados na Figura 7, são eles: a dimensão física dos *pixels* indicados por h_x e h_y, geralmente expressado em micrômetro (μm); a origem do sistema de coordenadas da imagem no canto esquerdo superior; o ponto indicado por $o(x_o, y_o)$ (que não representa a origem do plano Π), chamado de ponto principal ou central, formado a partir da projeção ortogonal da origem do sistema de câmera O_c em Π - plano xy do sistema bidimensional da imagem; o comprimento focal f, isto é, a distância entre O_c e o ponto principal o.

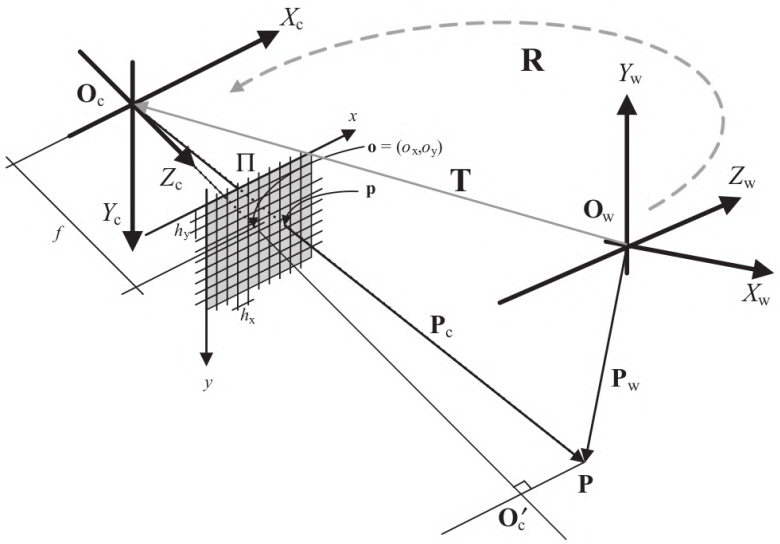
Em coordenadas cartesianas, as Equações 2.8 e 2.9 demonstram como a transformação entre o *frame* da câmera e da imagem é possível, dado um ponto $p(u, v)$ qualquer no plano Π.

$$u = h_u \frac{f}{z} x + u_o \tag{2.8}$$

$$v = h_v \frac{f}{z} y + v_o \tag{2.9}$$

A partir dos pontos em coordenadas homogêneas, é possível reescrever a Equação 2.7 considerando, agora, o conjunto completo dos parâmetros intrínsecos. Essa nova representação

Figura 7 – Representação dos parâmetros intrínsecos e extrínsecos de uma câmera.



Fonte: (CYGANEK; SIEBERT, 2011).

matricial é ilustrada na Equação 2.10.

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fh_u & 0 & u_o & 0 \\ 0 & fh_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.10}$$

2.3.3 Parâmetros Extrínsecos

Dado que os parâmetros extrínsecos estão relacionados aos elementos externos da câmeras, sua interpretação está intimamente relacionado ao sistema de câmera utilizado mas, em suma, são todos os parâmetros geométricos que permitem a mudança do sistema de coordenadas de mundo para o sistema de coordenadas de câmera e vice-versa.

Na Figura 7, o *frame* de mundo O_W e o de câmera O_C são destacados. O vetor T descreve uma mudança na posição dos centros de coordenadas dos sistemas O_W e O_C e a matriz R representa a rotação nos eixos correspondentes de cada sistema. Juntos, T e R constituem os parâmetros extrínsecos, representados pela matriz M da Equação 2.11.

$$[R \quad T] = M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{2.11}$$

A relação que garante a transformação de coordenadas de mundo para coordenadas

de imagem depende, portanto, das matrizes de parâmetros intrínsecos e extrínsecos, como mostrado na Equação 2.12.

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fh_u & 0 & u_o & 0 \\ 0 & fh_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.12}$$

Ou de uma forma mais sucinta, a transformação de $\widetilde{p}$ em coordenadas de imagem para sua respectiva coordenada de mundo $\widetilde{P}_W$, pode também ser indicada pela Equação 2.13, que leva em consideração a matriz de parâmetros intrínsecos K, e extrínsecos M, onde λ é o fator de escala.

$$\lambda\widetilde{p} = K[R \quad T]\widetilde{P}_W \tag{2.13}$$

2.3.4 Parâmetros de Distorção

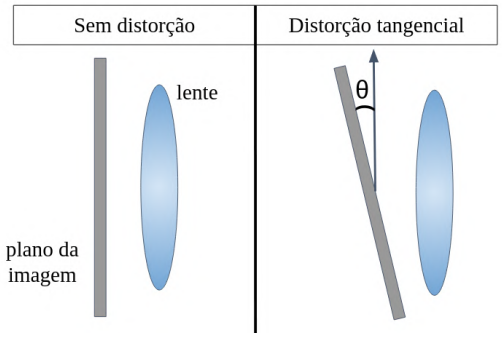
Num cenário ideal, linhas retas do mundo são representadas em linhas retas na imagem. No entanto, devido à curvatura da lente, as imagens resultantes sofrem algum tipo de distorção. Na verdade, existem duas distorções principais da lente: a radial e a tangencial. A primeira é causada pela lente em si, ilustrada na Figura 8, e a segunda resultante do não paralelismo da lente com o plano de imagem, como mostra a Figura 9. A completa modelagem da câmera também deve considerar o cálculo desses valores para que, assim, seja possível transformar as coordenadas da imagem distorcidas, em coordenadas ideais, removendo a distorção.

Figura 8 – Tipos de distorções radias: (a) negativa e (b) positiva. Por fim, (c) mostra a lente ideal sem distorções.

Fonte: (MATHWORKS, 2016).

As Equações 2.14 e 2.15 ilustram uma possibilidade de relação entre a imagem desejada (u, v) e a com distorção (u_d, v_d) radial e tangencial, respectivamente. Os elementos $r_{n(n=1,2,3)}$ denotam as distorções radiais, e $t_{n(n=1,2)}$ as tangenciais. Por fim, tem-se que d varia em função

Figura 9 – A distorção tangencial é caracterizada pelo não paralelismo entre o plano da imagem e a lente de convergência.



Fonte: autora.

da imagem sem distorção, como ilustrado na Equação 2.16, onde o ponto principal do plano de imagem é representado por (u_o, v_o).

$$\begin{aligned} u_d &= u(1 + r_1 d^2 + r_2 d^4 + r_3 d^6) \\ v_d &= v(1 + r_1 d^2 + r_2 d^4 + r_3 d^6) \end{aligned} \tag{2.14}$$

$$\begin{aligned} u_d &= u + 2t_1 uv + t_2(d^2 + 2u^2) \\ v_d &= v + 2t_2 uv + t_1(d^2 + 2v^2) \end{aligned} \tag{2.15}$$

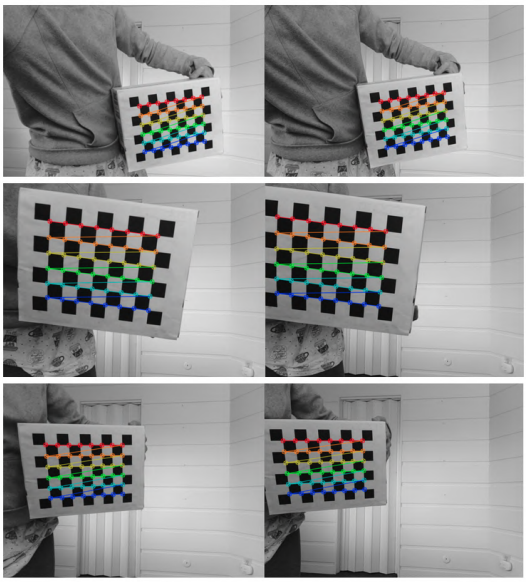
$$d = \sqrt{(u_d - u_o)^2 + (v_d - v_o)^2} \tag{2.16}$$

Dessa forma, os elementos de distorção fazem parte do conjunto de parâmetros intrínsecos da câmera. Seus valores são calculados a partir da etapa conhecida como calibração da câmera, abordada na Seção 2.3.5.

2.3.5 Calibração da Câmera

A etapa conhecida como *calibração da câmera* é realizada a fim de calcular os parâmetros intrínsecos e extrínsecos da câmera. Inicialmente, é preciso obter imagens de um padrão previamente conhecido. Assim, um algoritmo de processamento de imagem pode facilmente reconhecê-lo e, a partir da correspondência dos pontos 3D da cena com os 2D do plano, estimar os parâmetros. Neste trabalho, adotou-se o padrão planar com um tabuleiro de xadrez (ZHANG, 2000). O método detecta os cantos em múltiplas visualizações e estima os parâmetros que minimiza o erro entre as coordenadas do espaço e suas respectivas projeções nos planos de imagem. O erro mínimo é calculado a partir do algoritmo de Levenberg-Marquardt (MORÉ, 1978), que depende de uma primeira estimativa dos parâmetros. A Figura 10 ilustra apenas três pares de imagem do tabuleiro visto de diferentes perspectivas.

Figura 10 – Processo de calibração da câmera a partir de um padrão planar, neste caso, o tabuleiro de xadrez.



Fonte: autora.

A calibração por padrão planar (2D) é amplamente empregado a partir do tabuleiro de xadrez, mas há também a utilização de padrões 1D (BORGHESE; CERVERI, 2000) e até adimensional, com o uso de sensor infra-vermelho (SVOBODA; MARTINEC; PAJDLA, 2005). Quando há pouco controle sobre a configuração de imagem (por exemplo, com uma única imagem da cena), a calibração pode ser realizada através de métodos baseados em aprendizado profundo (LOPEZ et al., 2019).

Diversas técnicas propostas e disponíveis em código aberto, e.g biblioteca do OpenCV, juntamente com sua respectiva referência para maiores detalhamentos, podem ser acessadas em (BOUGUET, 2015). Uma visão mais detalhada sobre esse tópico, das técnicas e da matemática envolvida pode ser encontrado em (HARTLEY; ZISSERMAN, 2004).

2.4 CORRESPONDÊNCIA ESTÉREO

Como abordado na Seção 2.2, a segunda macro atividade para estimar a profundidade pelo método da triangulação, dado um sistema de câmera estéreo, é encontrar os pontos correspondentes de uma *imagem-base* em uma outra *imagem-correspondente*. Este conceito é conhecido como correspondência estéreo (em inglês, *Stereo matching*), a qual consiste numa linha de pesquisa bastante atual (FAN et al., 2020).

As buscas por pontos correspondentes entre duas imagens bidimensionais podem ser facilmente limitadas à uma linha a partir da chamada *restrição epipolar horizontal (ou vertical)*. Este conceito faz parte da *geometria epipolar*, e essa otimização nas buscas se dá por uma etapa de retificação das imagens. Dado que este trabalho fez uso dessa estratégia, a Seção

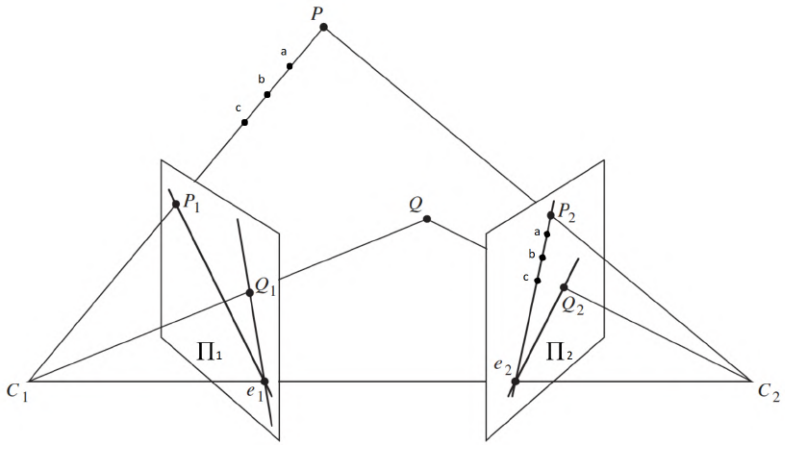
2.4.1 aborda tanto a geometria epipolar quanto a etapa de retificação. Em seguida, a Seção 2.4.2 trata sobre os tipos de correspondência estéreo.

2.4.1 Geometria epipolar

Geometria epipolar é a geometria voltada para um sistema estereoscópico. Ela é adotada pela necessidade de encontrar uma correspondência entre pontos que aparecem nas diferentes imagens de um mesmo cenário. As buscas por pontos correspondentes entre duas imagens pode ser facilmente reduzida à uma dimensão a partir do que se chama de *restrição epipolar horizontal (ou vertical)*. Pela Figura 11, é possível levantar algumas observações que auxiliarão no entendimento desse conceito, são elas:

- A origem do sistema de coordenadas da câmera esquerda C_1 e a origem do sistema de coordenadas da câmera direita C_2 formam o chamado *eixo epipolar*, e a distância entre essas duas origens determina o *baseline* da câmera;
- Os pontos P, C_1 e C_2 formam um *plano epipolar*, sendo P um ponto qualquer no espaço 3D. Infintos planos podem ser definidos uma vez que P mude de posição no espaço;
- Os dois pontos denotados por e_1 e e_2 são chamados de *epipolos*, ou seja, são pontos formados a partir da intersecção do eixo epipolar com os planos das imagens esquerda Π_1 e direita Π_2. Quando não há intersecção, considera-se que os pontos epipolares encontram-se no infinito;
- A linha formada pelo ponto epipolar e um ponto qualquer no plano da imagem forma a chamada *linha epipolar*, por exemplo $\overline{P_1 e_1}$ e $\overline{Q_1 e_1}$. Considerando que a imagem é usualmente apresentada em unidades de *pixel*, o número de linhas é finito.

Figura 11 – A restrição epipolar garante que o correspondente do ponto P_1 está na linha epipolar conjugada do plano Π_2.



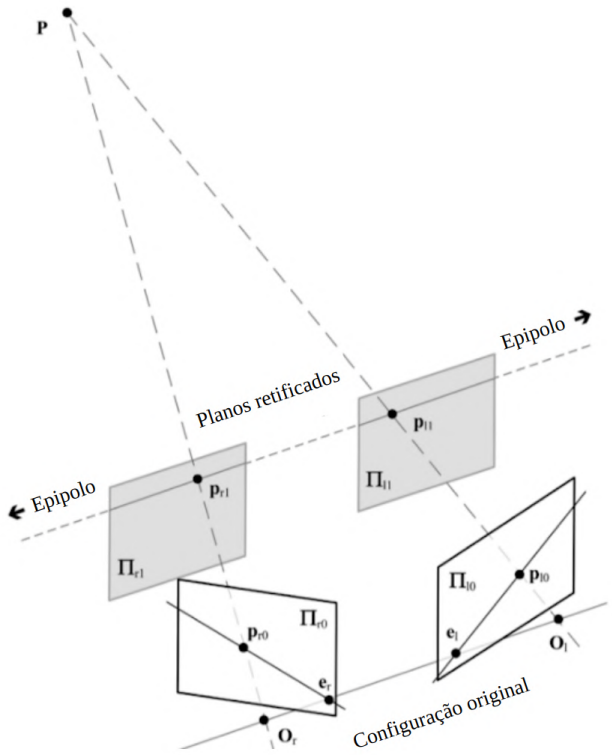
Adaptada: (HARTLEY; ZISSERMAN, 2004).

A partir disso, é importante observar que os pontos sob $\overline{PP_1}$, exemplificados por a, b e c, interceptam o plano Π_1 no mesmo ponto P_1 dado que estão diretamente alinhados com o centro de projeção C_1 da câmera. No entanto, esses mesmos pontos tem representações distintas no plano Π_2, também denotado por a, b e c. Dessa forma, tem-se que todo e qualquer ponto sob $\overline{PP_1}$ terá uma representação em Π_2 na linha epipolar $\overline{P_2e_2}$. Portanto, um ponto qualquer observado em uma imagem deve ser projetado sobre a linha epipolar correspondente na outra imagem. Esse conceito é chamado de *restrição epipolar*.

Partindo do pressuposto que os epipolos são conhecidos, a busca por pontos correspondentes envolve duas etapas: determinar a linha epipolar na imagem-base e, dentre os *pixels* da linha equivalente na imagem-correspondente, encontrar o melhor candidato para correspondência. A fim de eliminar a primeira etapa, o processo de retificação é requerido, garantindo assim uma *restrição epipolar horizontal ou vertical*. Essa restrição garante que um ponto $P_1(u, v)$ na imagem esquerda, tem seu correspondente na linha u da imagem direita (no caso de restrição horizontal).

A Figura 12 ilustra os planos de projeção Π_1' e Π_2' definidos a partir da etapa de retificação, os quais são coplanares entre si. Além disso, seus respectivos epipolos estão no infinito, resultando em linhas epipolares paralelas ao eixo epipolar. O detalhamento matemático da etapa de retificação foge da escolpo deste trabalho, mas maiores detalhamentos podem ser encontrados no livro (HARTLEY; ZISSERMAN, 2004).

Figura 12 – Os planos Π_1 e Π_2 representam a configuração original, enquanto os planos Π_1' e Π_2' são resultantes do processo de retificação do sistema estéreo.

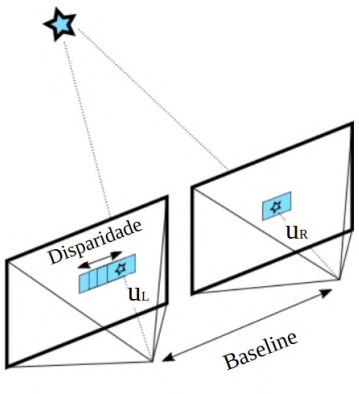


Adaptada: (CYGANEK; SIEBERT, 2011).

2.4.2 Tipos de Correspondência Estéreo

Na Seção 2.2, foi apresentada a Equação 2.6 a fim de mostrar como a profundidade Z poderia ser calculada a partir de determinados elementos, entre eles estão o f, que representa o comprimento focal, e o b, que indica o *baseline* da câmera. Ambos fazem parte do conjunto de parâmetros intrínsecos explanados na Seção 2.3.2. Considerando a imagem uma matriz 2D, as variáveis u_L e u_R representam as respectivas posições de coluna dos pontos correspondentes (assumindo que as imagens estão retificadas horizontalmente). Por convenção, a diferença dos valores u_L e u_R é denominada de *disparidade*, como ilustrado na Figura 13.

Figura 13 – Ilustração da disparidade: o objeto capturado por ambas as imagens tem uma distância relativa de d unidades de *pixels* na horizontal.



Fonte: autora.

Dessa forma, considerando um par de imagens retificadas e sem distorções, um ponto $p(i, j)$ na imagem-base tem seu equivalente na imagem-correspondente em $p(i, j - d)$, sendo d o valor de disparidade. A partir desse conceito, a equação que envolve a profundidade Z, é reapresentada pela Equação 2.17, sendo f o comprimento focal, b o *baseline*, e u_L e u_R a posição de pontos correspondentes.

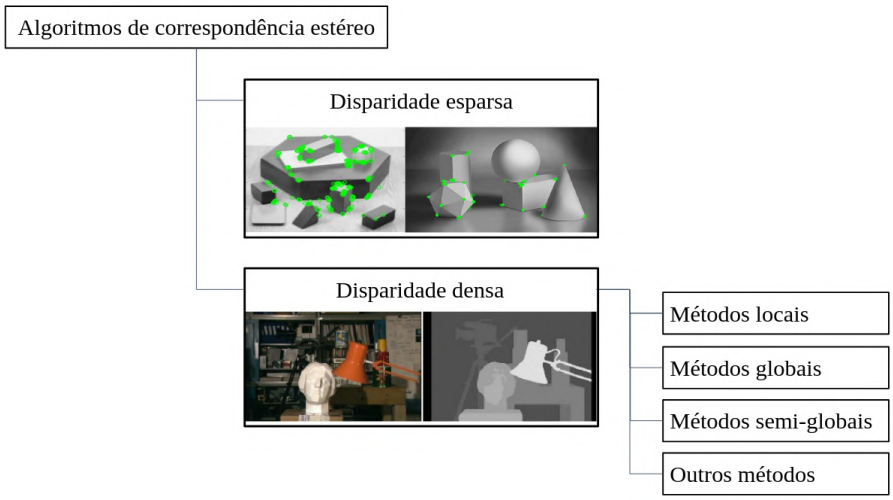
$$Z = \frac{fb}{u_L - u_R} = \frac{fb}{d} \tag{2.17}$$

Os algoritmos de correspondência estéreo, que buscam definir *pixels* equivalentes, resultam em uma imagem conhecida como *mapa de disparidade*. Usualmente, a imagem esquerda é a definida como referência. Assim, cada *pixel* do mapa indica quantas unidades de *pixel* devo deslocar a fim de encontrar o *pixel* correspondente na imagem direita (Figura 13). Este mapa pode ser dado de forma esparsa ou densa, ilustrado pela fluxo na Figura 14, onde, no primeiro caso, apenas os *pixels* destacados em verdes vão ter a informação de disparidade. Já no caso de mapa denso, cada *pixel* da imagem tem um respectivo valor de disparidade. Nesse trabalho, se faz o uso dos dois tipos de mapas:

- O mapa denso é a primeira informação requerida a fim de estimar a região livre para navegação. Portanto, as explanações da correspondência estéreo densa e do algoritmo que utilizamos para extrair o mapa denso, são abordadas nas Seções 2.5 e 2.6, respectivamente.

- O mapa esparso é utilizado pelo sistema SLAM de localização e mapeamento simultâneos, onde o foco de se obter uma localização robusta é maior do que uma reconstrução densa do ambiente. A forma como o sistema SLAM adotado adquire o mapa de disparidade é um dos itens abordados no Capítulo 3.

Figura 14 – Tipos de mapa de disparidade. O esparso calcula a disparidade apensas de pontos específicos (por exemplo, dos pontos destacados em verdes). Já o denso calcula a disparidade para cada *pixel* da imagem.



Fonte: autora.

2.5 MAPA DE DISPARIDADE DENSO

Com relação aos algoritmos de correspondência estéreo densa, mostrados na Figura 14, tem-se que os de métodos locais avaliam a similaridade entre *pixels* através de informação pontual ou dos arredores do *pixel*. Usualmente são alternativas mais rápidas, porém mais susceptíveis à resultados ruidosos e imprecisos. Por outro lado, os de métodos globais atribuem os valores de disparidade considerando a informação de toda a imagem. Seus resultados são mais precisos, no entanto, dependem de diversas iterações, aumentando sua complexidade computacional e consumo de tempo, podendo levar de segundos a minutos (LAZAROS; SIRAKOULIS; GASTERATOS, 2008). Essas duas abordagens podem ser associadas para formar algoritmos de método semi-global, onde existe tanto uma etapa com estratégia local como global. Dado que este é o caso usado no nosso trabalho, seus conceitos serão mais detalhados no decorrer dessa seção.

Para discorrer sobre essa linha de pesquisa, tomaremos como base a taxonomia proposta por Scharstein (SCHARSTEIN; SZELISKI, 2002), a qual classifica as etapas de um algoritmo qualquer de correspondência estéreo, seja ele local, global ou outros. É importante ressaltar que as etapas não são parte obrigatórias de todas as técnicas propostas. São elas:

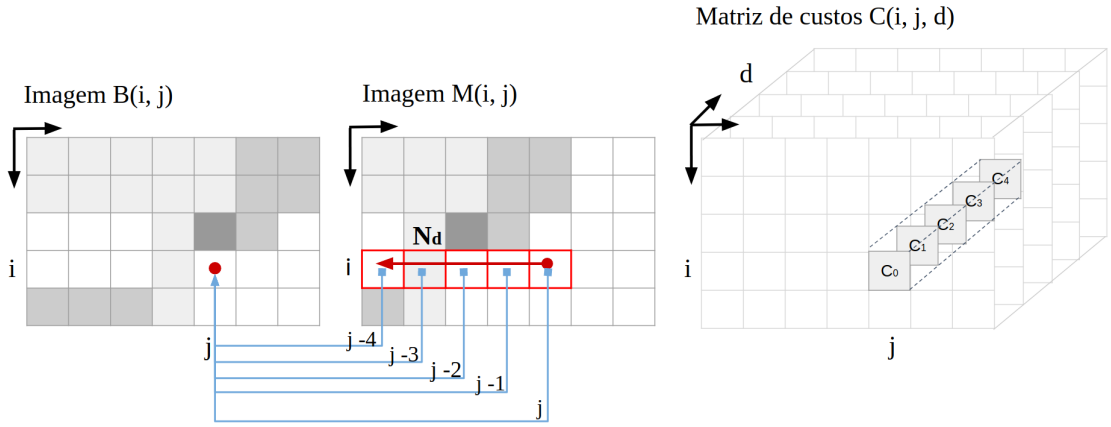
- Cálculo de custos da correspondência;
- Agregação dos custos;
- Cálculo da disparidade;
- Aperfeiçoamento da disparidade.

Para uma melhor compreensão, uma técnica de aquisição do mapa de disparidade, a *soma da diferença absoluta* (Sum of Absolute Differences (SAD)) , será ilustrada juntamente com a descrição de cada classificação.

2.5.1 Cálculo de Custos de Correspondência

O *custo* indica quão semelhante ou diferente são dois *pixels* entre si. Seus valores provêm da chamada *função de custo*, onde numa função assimilaridade, quanto menor o seu valor, maior o indicativo de semelhança. Seguindo com o exemplo da técnica SAD, esta utiliza como critério de semelhança a intensidade do *pixel*. A Equação 2.18 ilustra a função de custo conhecida como *diferença absoluta* (AD), onde I representa o valor da intensidade do *pixel* em questão. Esta equação, em conjunto com a Figura 15, mostra que determinado *pixel* $p(i, j)$ na imagem-base $B(i, j)$ é comparado com outros N_d *pixels* da imagem-correspondente $M(i, j)$, onde N_d representa a *faixa de disparidade* da busca, que neste caso, é igual à cinco. Repetindo esse processo para cada N_d *pixels* de $B(i, j)$, é gerada uma matriz tridimensional de custos $C(i, j, d)$, com dimensões de altura e comprimento da imagem, e o valor da faixa de disparidade.

Figura 15 – Par de imagens esquerda e direita; A matriz de custo tridimensional é preenchida a partir da função de custo adotada e da comparação de N_d *pixels*.



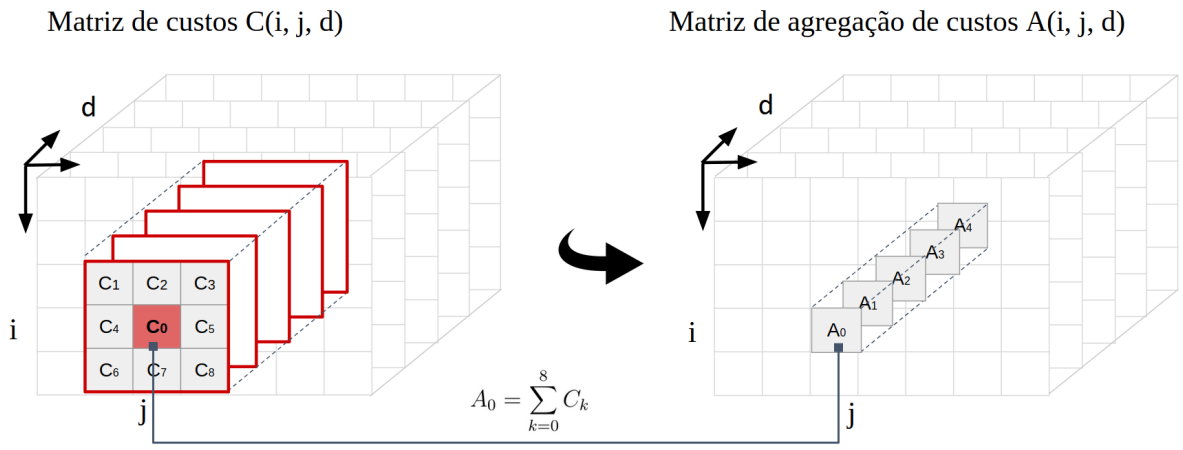
Fonte: autora.

$$C_d = C(i,j,d) = |I_B(i,j) - I_M(i,j-d)| \tag{2.18}$$

2.5.2 Agregação dos Custos

Após a obtenção de uma primeira matriz de custo $C(i,j,d)$, a etapa de agregação representa a forma com que esses custos vão ser interpretados e transformados em novos dados. A Figura 16 ilustra como a técnica SAD faz esse processo, onde o cálculo de uma iteração é mostrado. A agregação é feita somando os custos dentro de janelas de tamanho fixo (exemplificadas pelos quadrados com borda em vermelho), repetindo o processo em cada camada de d da matriz C. Com a repetição desse processo para cada valor de custo da matriz $C(i,j,d)$ uma nova matriz, a de agregação de custos $A(i,j,d)$, é formada.

Figura 16 – Ilustração de como a matriz de agregação é formada considerando a técnica SAD.



Fonte: autora.

É importante ressaltar que algoritmos de abordagens globais usualmente não apresentam esta etapa, substituindo-a por um procedimento de minimização de custo global, detalhada na Seção 2.5.3, que leva em consideração todo o mapa de disparidade.

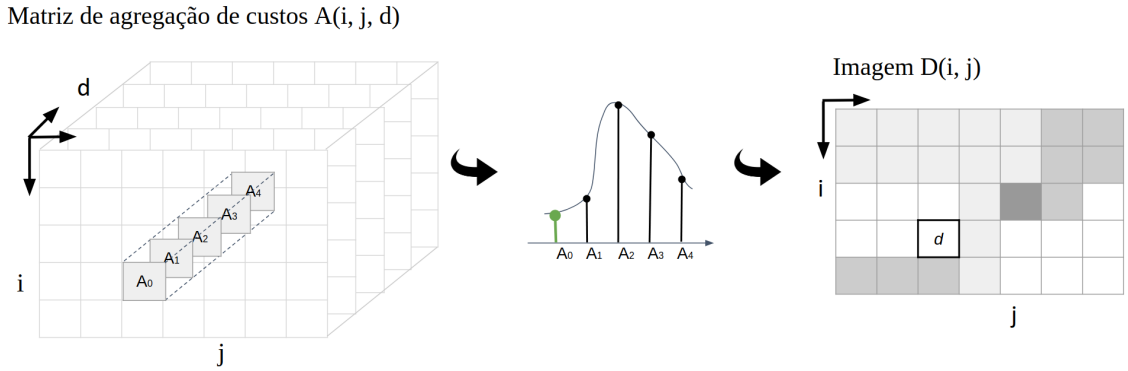
2.5.3 Cálculo da Disparidade

A etapa do cálculo de disparidade, no que diz respeito às estratégias locais, são executadas tomando a disparidade associada ao valor mínimo (ou máximo para função de similaridade) do vetor $A(i,j,k)_{k=1,2...N_d}$. Dessa forma, apenas o valor k de disparidade é variado, como representado na Equação 2.19.

$$D(i,j) = \min_{k\epsilon[0,N_d-1]} A(i,j,k) \tag{2.19}$$

A Figura 17 traz uma representação visual dessa etapa, e é importante observar que a matriz de custo envolvida nesse processo depende da técnica adotada, podendo ser a matriz inicial ou de agregação. Considerando a técnica SAD, a ilustração refere-se a matriz de agregação de custos $A(i,j,d)$. O resultado é uma imagem 2D em escala de cinza, isto é, o mapa de disparidade. Nele, quanto maior o valor de disparidade, mais escuro é o *pixel*.

Figura 17 – Ilustração de como o mapa de disparidade é formado a partir da matriz de custos inicial ou de agregação e de uma função de assimilaridade.



Fonte: autora.

Abordagens de métodos globais executam essa etapa através de diversas iterações em busca de uma minimização da função de custo global que leva em conta toda a imagem. Tal função possui um termo de energia dos dados e outro de suavidade, como apresentado na Equação 2.20, onde λ é um parâmetro de entrada que equilibra a influência dos fatores e define o grau de suavização do mapa de forma que quanto menor este valor, mais suavizado é o resultado. Esta equação 2.20 é iterada até o momento em que se encontra o menor valor da função de energia global.

$$E(d) = E_{dado}(d) + \lambda E_{suavidade}(d) \tag{2.20}$$

O fator de energia de dados, apresentada na Equação 2.21, mede quão bem está a atribuição das disparidades. Quanto menor seu valor, melhor. I_B denota todos os *pixels* da imagem-base e $C(p, p')$ é a função de custo que relaciona um *pixel* $p(i, j)$ da imagem-base e $p'(i, j - d)_{d=0,1...N_d-1}$ da imagem-correspondente. Esse cálculo, por exemplo, poderia ser realizado pela função AD, de diferença absoluta dos valores de intensidade.

$$E_{dado}(d) = \sum_{p \epsilon I_B} C(p, p') \tag{2.21}$$

O fator de energia de suavização parte do pressuposto que uma superfície não apresenta descontinuidades bruscas, de forma que a equação relacionada ao fator $E_{suavidade}(d)$ penaliza disparidades abruptas entre os *pixels* vizinhos, denominados por p e q na Equação 2.22; N denota todos os pares de *pixels* vizinhos na imagem-base. A função $s()$ atribui uma penalidade se a disparidade de p for (muito ou pouco) diferente daquela de q. Dessa forma, a penalidade resulta no aumento de energia de suavidade.

$$E_{suavidade}(d) = \sum_{<p,q> \epsilon N} s(D(p), D(q)) \tag{2.22}$$

A influência da energia de dados e de suavidade pode ser observada pela Figura 18. Os mapas de disparidade são resultados de diferentes configurações de λ na Equação 2.20.

É possível pontuar que, se definirmos $\lambda = 0$, ou seja, desconsiderar o termo de suavidade, o resultado do método global se resume a um método puramente local.

Figura 18 – Resultado do mapa de disparidade para valores crescentes do termo λ da Equação 2.20.

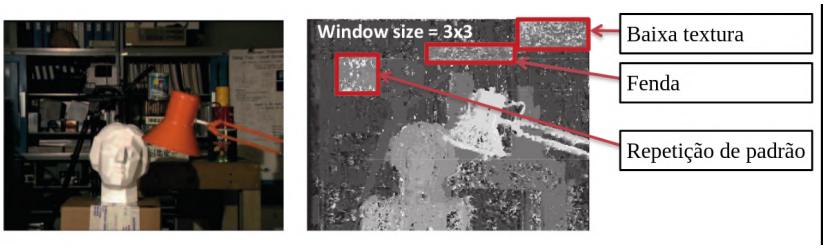
Fonte: (BLEYER; BREITENEDER, 2013).

2.5.4 Aperfeiçoamento da Disparidade

A etapa de aperfeiçoamento do mapa de disparidade (*Disparity refinement*) é adotada como forma me melhorar o resultado preliminar. A maioria dos métodos de refinamento segue o esquema de detecção das possíveis falhas, e seu respectivo preenchimento, seguido por uma etapa de filtragem (YAN et al., 2019). Existe uma vasta gama de trabalhos que contemplam e exploram esta etapa, como exemplificado em (HAMZAH; IBRAHIM, 2016).

As falhas são causadas por situações que dificultam a correta associação dos *pixels*, por exemplo, a diferença de iluminação entre os *frames*, chamada de diferenças radiométricas. Isto porque mesmo câmeras do mesmo modelo apresentam diferenças sutis, desde a lente até os sensores sensíveis a luz. O trabalho (HIRSCHMULLER; SCHARSTEIN, 2008) traz uma avaliação detalhada de como esse fator pode impactar os resultados. Outro desafio é causado por regiões de baixa textura, dessa forma, a correspondência sofre com a ambiguidade. A lista de desafios também inclui estruturas repetidas, isto é, a repetição de um padrão; objetos translúcidos ou transparentes; regiões de oclusão, ou seja, quando uma parte da cena é visualizada por uma imagem e não é pela outra; e descontinuidades abruptas e profundas. A Figura 19 mostra o mapa de disparidade gerado a partir da técnica local SAD com janela de tamanho 3×3. Nela é possível visualizar os resultados ruidosos devido à baixa textura, da descontinuidade abrupta, e da repetição de padrão.

Figura 19 – Imagem esquerda seguida do mapa de disparidade o qual é resultante da técnica SAD com janela de tamanho 3×3. Dificuldades encontradas, como baixa textura, são destacadas no mapa.



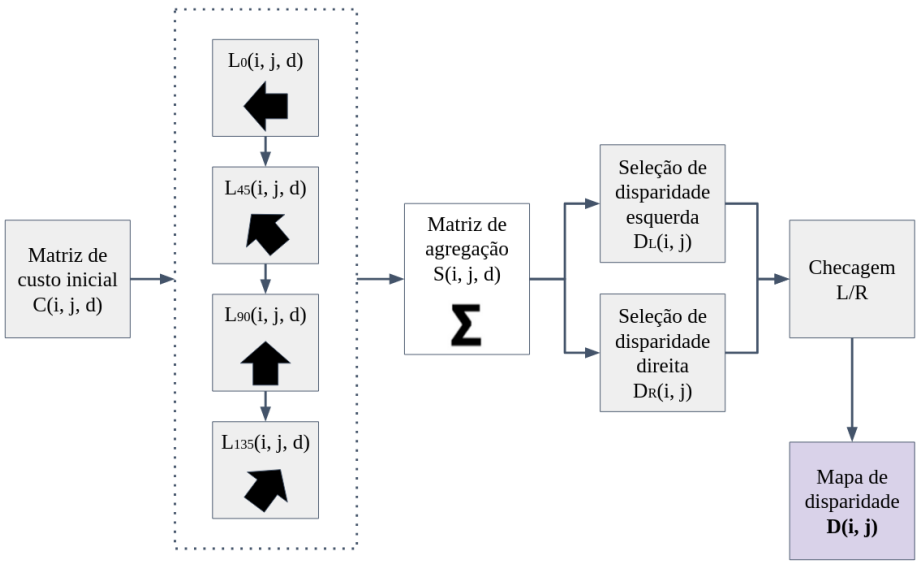
Fonte: (BLEYER; BREITENEDER, 2013).

2.6 CORRESPONDÊNCIA SEMI-GLOBAL - SGM

O algoritmo de correspondência estéreo utilizada neste trabalho é o de Correspondência Semi-global (Semi Global Matching (SGM)) (HIRSCHMULLER, 2008). Essa técnica realiza uma otimização em toda a imagem, produzindo mapas de disparidade mais precisos para o contexto real. Além disso, apresenta rapidez de processamento semelhante às de abordagens locais, porém com resultados mais aproximados de técnicas globais, ou seja, mais precisos.

A Figura 20 ilustra o fluxograma do SGM. Inicialmente é definida uma matriz de custo $C(i, j, d)$. Em seguida, as matrizes $L_{r^t}(i, j, d)$ começam a serem preenchidas, onde o cálculo dos seus elementos considera um único sentido r^t. A matriz de agregação $S(i, j, d)$ é definida pela soma matricial das matrizes da etapa anterior uma vez que todas são preenchidas. Com isso, a seleção de disparidade para o par da imagem é realizada, seguida da checagem L/R. Por fim, chega-se ao mapa de disparidade $D(i, j)$.

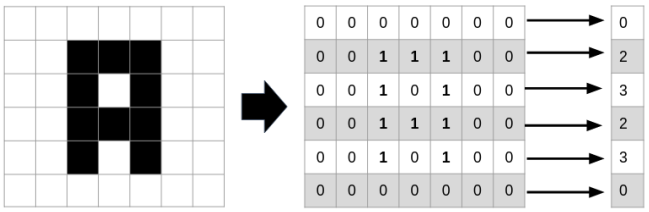
Figura 20 – Fluxograma da técnica SGM.



Adaptada: (HIRSCHMULLER, 2008).

Considerando a taxonomia apresentada na Seção 2.5, tem-se que a técnica SGM, na etapa de *cálculo de custos de correspondência*, obtém a matriz de custo inicial $C(i, j, d)$ a partir da entropia das imagens. A Figura 21 ilustra de forma simples o conceito de entropia no que diz respeito à imagem. A matriz binária é apresentada, onde *pixels* pretos são indicados por 1, e brancos por 0. Considerando uma avaliação por linha, a entropia da mesma será definida pelo número de vezes que há uma mudança de valor. Como exatamente o cálculo de $C(i, j, d)$ é realizado por meio da entropia foge do escopo desse trabalho, mas seu detalhamento pode ser encontrado em (VIOLA; III, 1997).

Figura 21 – Ilustração de como o conceito de entropia é aplicado à imagem.



Fonte: autora.

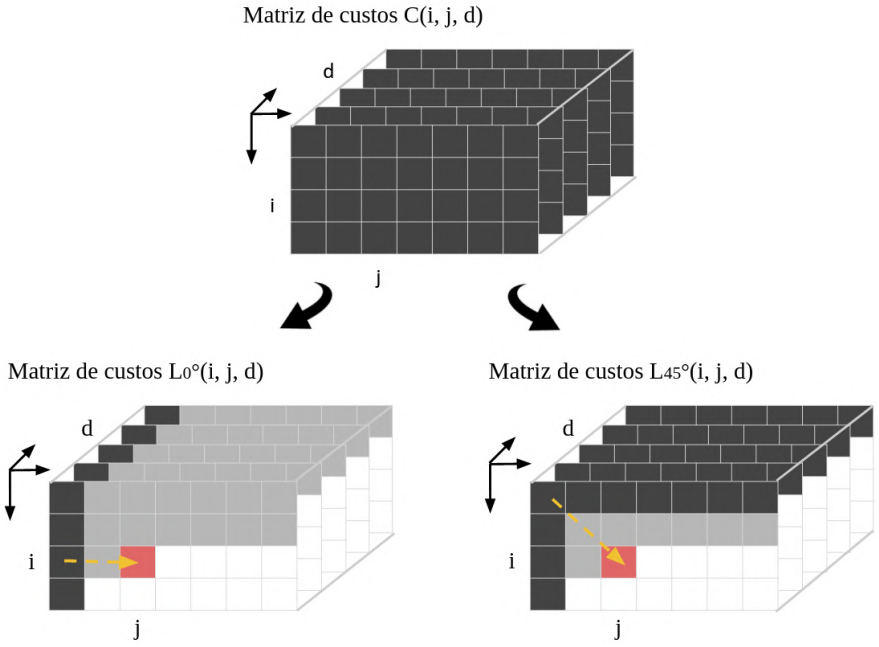
Uma vez que a matriz de custo inicial é definida, matrizes secundárias $L_{r^t}(i, j, d)$ podem ser formadas. A Figura 22 mostra uma matriz de custo C e duas matrizes secundárias L, onde à priori suas bordas são preenchidas com valores provenientes de C, indicados por cinza escuro. Os quadrados em cinza claro são elementos calculados anteriormente de forma recursiva (a partir dos de borda). As setas tracejadas indicam que o cálculo do elemento da iteração atual, em vermelho, deve considerar um sentido específico. Por fim, os elementos em branco indicam custos que ainda vão ser definidos.

O significado do sentido, indicado por r^t, é ilustrado na Figura 23. Por exemplo, para $r^t = 0^o$, o elemento $L_{0^o}(i, j, 1)$, destacado em vermelho, tem seu cálculo diretamente relacionado aos elementos destacados em verde, ou seja, $L_{0^o}(i, j-1, k)_{k=0,1,2}$. Já no caso de $r^t = 45^o$, o cálculo de $L_{45^o}(i, j, 1)$ vai depender dos elementos em verde $L_{45^o}(i-1, j-1, k)_{k=0,1,2}$. Portanto, fica visível que o sentido determina quais os elementos que vão ser considerados para o cálculo do custo em questão.

Um parâmetro importante do SGM é justamente a quantidade de sentidos. Isso porque o algoritmo gera a partir de $C(i, j, d)$ tantas matrizes $L_{r^t}(i, j, d)$ quantas forem as direções r^t, o que impacta diretamente no custo computacional. Outro fator que também impacta nesse quesito é o número de disparidade N_d, o qual interfere diretamente no tamanho das matrizes de custos. Por exemplo, a matriz inicial é representada por $C(W, H, N_d)$, onde W e H representam o comprimento e a altura da imagem, respectivamente, e N_d a profundidade de C.

A Equação 2.23 formaliza como exatamente o valor dos elementos da matriz $L_{r^t}(p, d)$ são definidos, onde p indica o par de pontos (i, j) e r a direção/ângulo. Por exemplo, para

Figura 22 – Os elementos em cinza escuro das matrizes $L_{r^t}(i,j,d)$ são provenientes da matriz $C(i,j,d)$; os em cinza claro foram anteriormente calculados de forma recursiva a partir dos de borda; elementos em vermelhos são calculados na iteração atual e dependem do sentido em questão, isto é, 0^o ou 45^o.



Fonte: autora.

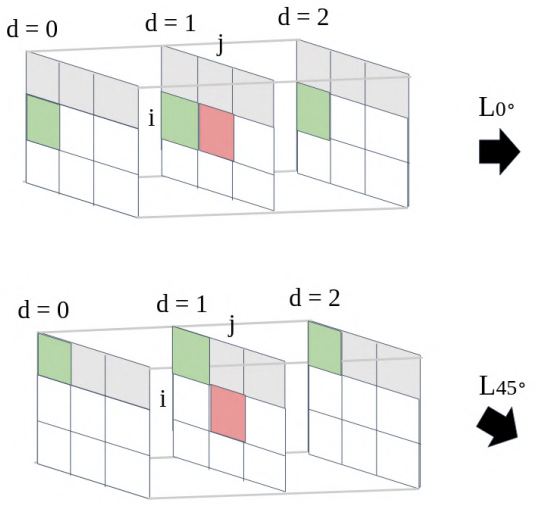
o cálculo de uma posição arbitrária $p = (i,j)$ com $d = 1$ e ângulo $r = 0^o$, teríamos que $L_r(p+r, d-1) = L_r(i, j-1, d-1)$. Caso $r = 45^o$, então, $L_r(p+r, d+1) = L_r(i-1, j-1, d+1)$. A Figura 23 contribui para essa visualização.

$$\begin{aligned} L_{r^t}(p,d) = min[L_r(p+r,d), L_r(p+r,d-1) + P_1, \\ L_r(p+r,d+1) + P_1, \\ \min_{i\epsilon[0,N_d-1]} L_r(p+r,i) + P_2] \end{aligned} \tag{2.23}$$

Pela equação, é possível perceber que $L_{r^t}(p,d)$ é definido pelo valor mínimo entre quatro termos. Quanto maior o valor de P_1 ou P_2, menores as chances dos termos em que eles aparecem de serem selecionados. Seus valores são estrategicamente alocados de forma que P_1 penalize pequenos saltos na disparidade, enquanto $P2$ penalize saltos mais significativos, dado que esses representam descontinuidades abruptas - mais improvável numa situação real. Pequenos saltos podem indicar uma superfície inclinada, sendo uma situação possível, diferentemente dos saltos abruptos, por isso, $P2 > P1$. É importante citar que quando algum dos quatro termos da equação não existe, lhe é atribuído um valor muito alto, de forma que ele não afete na escolha do valor mínimo. Como exemplo, o termo $P(i,j,d-1) = \infty$ no cálculo do elemento $P(i,j,0)$.

Ao finalizar a etapa de *agregação de custos*, a matriz final $S(i,j,d)$ é calculada pela

Figura 23 – O cálculo do custo do elemento em vermelho depende dos elementos em verde, cujo o ângulo é responsável por determiná-los.



Fonte: autora.

soma matricial de todas as matrizes de agregação $L_{r^t}(p,d)$ geradas no processo anterior, como mostra a Equação 2.24.

$$S(i,j,d) = \sum L_{r^t}(p,d) \tag{2.24}$$

Em seguida, na etapa final de *cálculo de disparidade*, é formada a matriz $D(i,j)$, onde cada elemento é igual ao valor de d que está associado ao custo mínimo do vetor $S(i,j,k)_{k=0,1...N_d}$. A Equação 2.25 formaliza esse cálculo.

$$D(i,j) = \min_{k\epsilon[0,N_d-1]} S(i,j,k) \tag{2.25}$$

Portanto, a matriz $D(i,j)$ é uma imagem com valores que variam entre 0 e N_d. A interpretação desse dado revela a distância, em unidades de *pixel*, entre elementos equivalentes da imagem-base e da imagem-correspondente, como explanado na Seção 2.4.2 e ilustrado pela Figura 13. Vale relembrar também que, de posse dos parâmetros da câmera e do mapa de disparidade $D(i,j)$, é possível calcular a profundidade Z para cada *pixel*, isto é, de forma densa (Equação 2.17).

3 SLAM BASEADO EM CARACTERÍSTICAS

Sistemas de SLAM visual (vSLAM) se utilizam de informação de imagens para se localizar e mapear simultaneamente. Como abordado no Capítulo 1, a forma como as imagens são interpretadas dividem o vSLAM em método direto ou indireto. Neste capítulo, trataremos apenas do segundo caso uma vez que é o sistema adotado neste trabalho. Além do mais, vale relembrar que a definição dos parâmetros do sistema estereoscópico (Seção 2.3) e o problema de correspondência estéreo (Seção 2.4) são assuntos fundamentais para o entendimento deste capítulo.

O primeiro conceito trazido é com relação ao termo *característica*, em inglês, *feature*. Uma *feature* é um pixel cujo seu entorno possui características bem definidas, facilmente reconhecidas e únicas. Idealmente, devem ser provenientes de pontos estacionários do ambiente. Bons candidatos, por exemplo, são regiões de cantos (em inglês, *corners*), onde se percebe uma mudança brusca de intensidade entre os *pixels* em mais de uma direção. A vantagem dos métodos baseados em *features* é a sua velocidade de processamento associada a um método de localização que já se provou bastante robusto (MUR-ARTAL; MONTIEL; TARDOS, 2015). Todavia, possuem a desvantagem de gerar um mapa esparso do ambiente, reduzindo a percepção do ambiente para o agente robótico.

O sucesso dessa abordagem depende de alguns pré-requisitos com relação ao ambiente: possuir suficiente iluminação; haver prevalência de mais objetos estáticos que móveis; existir textura suficiente para permitir extração de *features*; e existir suficiente sobreposição da cena capturada entre uma imagem e outra (para realizar o rastreamento das *features* extraídas).

3.1 BREVE HISTÓRICO

Em 2007, o algoritmo proposto por Georg Klein e David Murray (KLEIN; MURRAY, 2007), nomeado de Parallel Tracking and Mapping (PTAM) - *Parallel tracking and mapping*, trouxe uma solução que, como o nome sugere, paraleliza a etapa de rastreamento e mapeamento. Essa adaptação em específico representa um marco e um padrão de estruturação adotado pelas abordagens SLAM baseado em *features* que se sucederam (TAKETOMI; UCHIYAMA; IKEDA, 2017). Uma outra contribuição notória foi a adoção da técnica conhecida como *Bundle Adjustment*, responsável por refinar as estimativas de pose[1] e dos pontos 3D do mapa, tudo em um processamento de tempo-real. No entanto, o sistema proposto era voltado para aplicações de realidade aumentada (RA) e com limitação de mapeamento para pequena escala, por exemplo, do tamanho de uma mesa de escritório. O esforço da comunidade, portanto, voltou-se ao mapeamento em larga escala.

[1] É chamada de *pose* a informação de posição e orientação de um objeto, neste caso, a câmera, em função de determinado sistema de coordenadas. Em uma espaço tridimensional apresenta 6 graus de liberdade.

Em 2015, Mur-Artal et al. reuniu diversos avanços nas pesquisas descritas em (KLEIN; MURRAY, 2007; RUBLEE et al., 2011; KÜMMERLE et al., 2011; GÁLVEZ-LÓPEZ; TARDOS, 2012) no seu trabalho conhecido como ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015). Assim como no PTAM, Mur-Artal et al. também utilizou processamento em paralelo, totalizando três módulos: de rastreamento; de mapeamento; e o terceiro voltado para o reconhecimento de uma revisita e otimização dos dados estimados (nuvem de pontos e poses). Outra vantagem consiste na automática inicialização do mapeamento, uma vez que o PTAM requeria uma operação manual, posicionando a câmera numa distância conhecida de uma cena plana. Além do mais, Mur-Artal et al. adotou o detector e descritor de *features* Oriented FAST and rotated BRIEF (ORB), que possui uma taxa de processamento duas ordens de magnitude mais rápida que técnicas relacionadas (RUBLEE et al., 2011). Outro fator que corrobora com a qualidade do mapeamento e de seu processamento em tempo real é a estruturação de estimativas de poses da câmera e dos pontos 3D em grafos.

A primeira versão do projeto de código aberto ORB-SLAM, que em sua funcionalidade contemplava a câmera monocular, foi estendida, em 2017, para câmeras estéreo e Red, Green, Blue, Depth (RGB-D). No trabalho nomeado de ORB-SLAM2 (MUR-ARTAL; TARDÓS, 2017a), os autores apresentaram melhorias que resultaram em uma localização mais robusta que o sistema vSLAM estado da arte de método direto, o LSD-SLAM, proposto por (ENGEL; SCHÖPS; CREMERS, 2014). Uma outra contribuição notória foi a adição do modo "localização", que parte do pré-suposto que a região já foi mapeada. Nesse modo, o mapa é carregado e, uma vez que a câmera é re-localizada, o sistema continuamente estima somente a posição da câmera (sem adicionar novos pontos ao mapa). É uma funcionalidade interessante principalmente para ambientes mais estacionários.

Desde a primeira versão do ORB-SLAM, diversos trabalhos tomaram como base seu código aberto para estender suas funcionalidades (MUR-ARTAL; TARDÓS, 2017b) (CAMPOS et al., 2021) (GOMEZ-OJEDA et al., 2019) (PUMAROLA et al., 2017) ou simplesmente criar um novo vSLAM, como é o caso do OpenVSLAM, sistema adotado neste trabalho, o qual conta com bibliotecas próprias, responsáveis por otimizações comprovadas em (SUMIKURA; SHIBUYA; SAKURADA, 2019). Além disso, o OpenVSLAM foi elaborado para ser usado como um *framework* de um sistema SLAM baseado em *features*, de fácil entendimento e desenvolvimento para novos colaboradores. Em uma avaliação comparativa com ORB-SLAM3 e RTAB-Map, apresentada no trabalho descrito em (MERZLYAKOV; MACENSKI, 2021), os autores concluíram que o OpenVSLAM representa a melhor técnica de uso geral, isto é, para uma gama de tipos de robôs, ambientes e sensores de visão. Ainda acrescentou sobre seu bom desempenho nos ambientes avaliados, apresentando uma relocalização superior mesmo diante da variação de iluminação do ambiente.

É importante mencionar que, além das abordagens geométricas, mais recentemente, estratégias com *deep learning* tem sido exploradas como forma de localizar e re-localizar o agente. Um detalhamento sobre o assunto, de forma a explorar desafios e oportunidades de

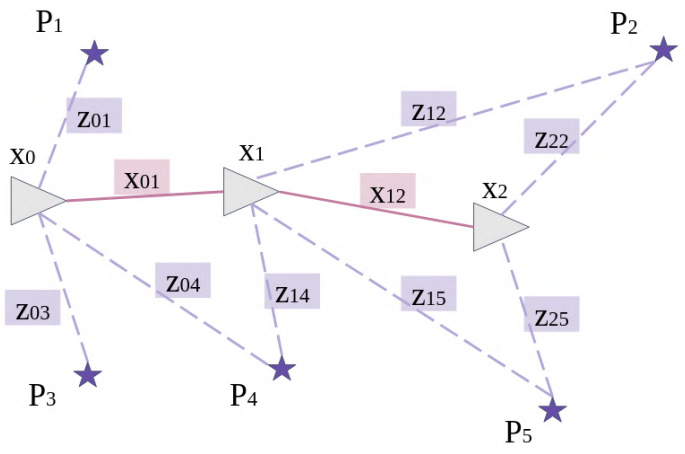
sistemas vSLAM com inteligência artificial, pode ser encontrado no trabalho desenvolvido em (LI; WANG; GU, 2018).

3.2 OPENVSLAM

A arquitetura de um sistema SLAM inclui dois componentes principais: o *front-end* e o *back-end*. O primeiro lida diretamente com os dados de entrada do sensor, neste caso, quadros de imagem, enquanto que o *back-end* é responsável por otimizar as interpretações produzidas pelo *front-end*.

Um fator que corrobora com o processamento em tempo real dessa arquitetura é a estruturação dos dados em grafo, inicialmente proposto em (LU; MILIOS, 1997), garantindo interdependência entre as variáveis. Pela Figura 24, é possível observar que um nó pode corresponder a uma estimativa de pose x ou a um ponto 3D P (proveniente do mapeamento de uma *feature*). Uma *aresta* entre dois nós representa uma *restrição* espacial (em inglês, *constraints*). Entre dois nós de poses, o comprimento da aresta indica a estimativa de odometria. É importante ressaltar que as restrições não são rígidas ou fixas. Elas podem ser interpretadas como molas, que são moldados a partir de estratégias de otimização do grafo. Uma outra observação a ser feita é que não há limites sobre o número de arestas entrando ou saindo de um nó.

Figura 24 – Dados do sistema SLAM estruturados em grafos de poses x e pontos 3D P.

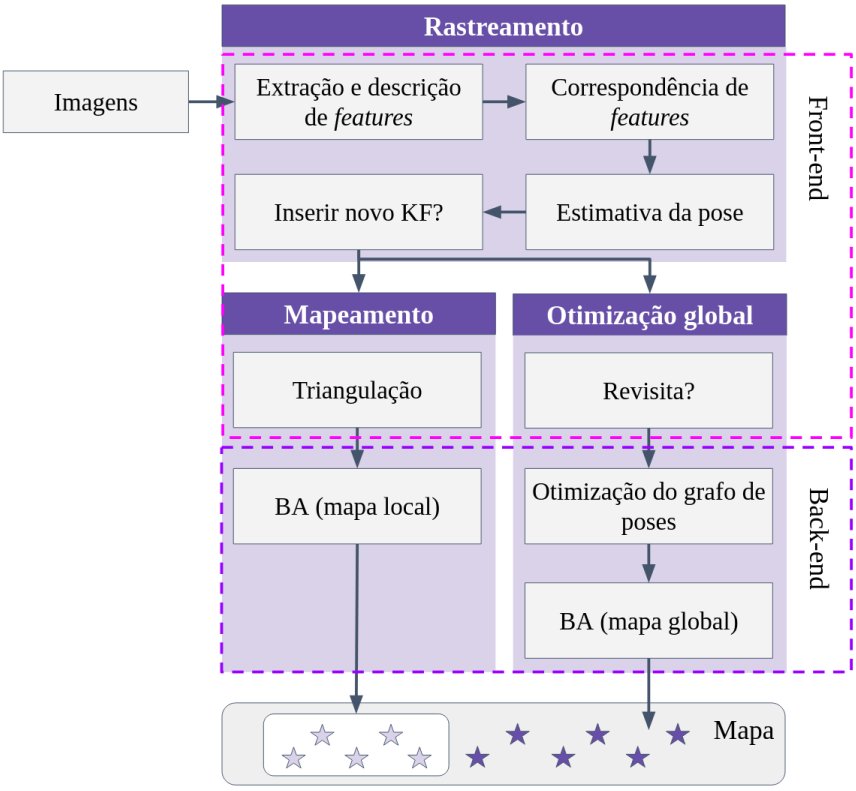


Fonte: autora.

Uma vez apresentado os conceitos de *front-end* e *back-end* de um sistema SLAM, e a estruturação dos dados, é possível aprofundar as etapas do OpenVSLAM, ilustradas na Figura 25. O fluxograma apresenta três módulos que são processados de forma paralela. No módulo de *rastreamento*, quando o par de imagens entra no sistema, ocorre a extração e descrição de *features*, seguido da correspondência das mesmas com pontos 3D do mapa. Dessa forma, uma primeira estimativa da pose da câmera pode ser dada. O módulo de rastreamento é executado

cada vez que uma nova imagem chega. Já os módulos de mapeamento e otimização global só são executados no momento em que se é definido um novo *keyframe*[2] (KF). Se uma revisita for detectada, (*loop closure*, em inglês), é executado uma otimização do grafo levando em consideração apenas as poses estimadas, e em seguida, uma otimização global considerando todos os dados, chamada de Bundle Adjustmente (BA) . O mapeamento, no entanto, é responsável por adicionar novos pontos 3D ao mapa através da correspondência de *features* entre o par de imagens e da triangulação (abordada na Seção 2.2.). Em seguida, executa localmente a técnica de BA.

Figura 25 – Três módulos são executados paralelamente durante o processo do sistema SLAM: rastreamento, responsável por extrair, descrever e encontrar *features* correspondentes, bem como uma primeira estimativa da pose da câmera; mapeamento, responsável por executar a triangulação e definir pontos 3D do mapa esparso do sistema, assim como executar o BA no mapa local; otimização global, responsável por reconhecer uma revisita e otimizar as estimativas de pose e dos pontos 3D no mapa global.



Adaptada: (SUMIKURA; SHIBUYA; SAKURADA, 2019), (MUR-ARTAL; TARDÓS, 2017a).

3.3 CRIAÇÃO DO GRAFO: *FRONT-END*

As etapas relacionadas à criação do grafo começa pelo módulo de rastreamento, trazido da Figura 25. Primeiro são extraídas e descritas as *features* da imagem; depois é realizada a

[2]Quadros-chave, ou em inglês, *keyframes*, são imagens específicas extraídas da sequência de imagens, definidas a partir de uma série de condições, detalhadas na Seção 3.3.4.

correspondência dessas *features* com pontos 3D do mapa local e a pose da câmera é estimada; por fim, é decidido se o quadro em questão será incorporado como *keyframe* no sistema SLAM. Se sim, o módulo de mapeamento executa a triangulação para gerar novos pontos 3D à nuvem esparsa e, no módulo de otimização global, é feito a verificação de uma possível revisita.

3.3.1 Detecção e Descrição de *Features*.

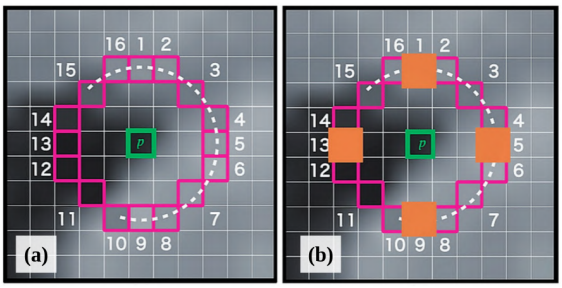
A detecção e descrição das *features* utilizados pelo OpenVSLAM se dá pela estratégia apresentada no trabalho (RUBLEE et al., 2011), em que as *features* são invariante à escala e à orientação (em inglês, *Oriented FAST and rotated BRIEF* - ORB). Cada uma delas é associada a um descritor de 256 *bits*, o que garante uma técnica extremamente rápida para computar e comparar as *features*. O bom desempenho da estratégia, quando adotada num sistema SLAM, é demonstrado em (MUR-ARTAL; TARDÓS, 2014). A proposta consiste em uma fusão do detector FAST (ROSTEN; DRUMMOND, 2006) com o descritor BRIEF (CALONDER et al., 2010), onde melhorias foram adicionadas. Sendo assim, FAST e BRIEF serão discutidos a seguir como forma de introdução ao ORB.

3.3.1.1 Detector FAST

O algoritmo FAST, *Feature from the Accelerated Segment Test*, toma um pixel p de intensidade I_p, e compara a intensidade de outros dezesseis *pixels* em seu entorno, como ilustrado na Figura 26.(a). Dessa forma, p é considerado uma *feature* se um conjunto de n *pixels* apresentarem uma intensidade maior do que $Ip + t$, ou menor que $Ip - t$, onde t é um valor limiar, e n é usualmente adotado como 12.

Para acelerar o método proposto, a comparação é realizada inicialmente nos *pixels* de índices 1, 5, 9 e 13, conforme destacado na Figura 26.(b). Se pelo menos três das quatro intensidades estiverem acima ou abaixo de $Ip \pm t$, o teste se estende para os outros *pixels*, caso contrário, o *pixel* candidato é rejeitado como *feature*.

Figura 26 – Reconhecimento de uma *feature* (a) a partir da comparação de um *pixel* p com outros dezesseis *pixels* no seu entorno; ou (b) com apenas quatro *pixels*, acelerando o processo do método FAST.



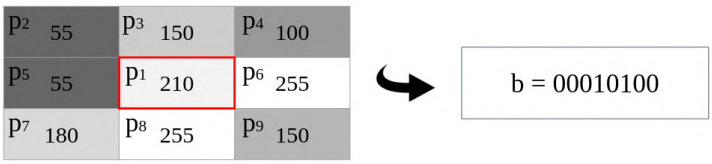
Fonte: (GAKNE; O'KEEFE, 2018).

3.3.1.2 Descritor BRIEF

A vizinhança em torno de um *pixel* é conhecida como bloco (*patch*, em inglês), isto é, um quadrado com determinada largura e altura. Uma vez definidos quais *pixels* que representam *features*, seus respectivos blocos podem ser descritos a partir de comparações de intensidade entre pares de *pixels*. Como forma de aumentar a robustez do descritor, a imagem é suavizada por meio de Kernel Gaussiano antes da aplicação da técnica BRIEF. Nela, os blocos de imagem são descritos como um vetor binário, onde seu tamanho pode variar de 128 à 512 *bits*. A Figura 27 em conjunto com a Equação 3.1 ilustra um exemplo prático. Tomando um par de *pixels*, tem-se que se $I(p_i) < I(p_j)$, onde I indica a intensidade, o valor concatenado no vetor de *bits* será 1, caso contrário, será 0. Como exatamente a seleção dos pares ocorre, foge do escopo deste trabalho, mas maiores detalhes podem ser encontrados no trabalho em (CALONDER et al., 2010), onde são descritas e avaliadas cinco estratégias possíveis.

$$b = \begin{cases} 1, & \text{se } I(p_1) < I(p_2); \\ 0, & \text{caso contrário.} \end{cases} \tag{3.1}$$

Figura 27 – Exemplo prático de um descritor BRIEF. Neste caso, apenas para fins ilustrativo, a intensidade do pixel central é comparado com todos os outros. Neste caso, a comparação é de p_1 com $p_{n(n=2,3...9)}$, nesta ordem.



Fonte: autora.

3.3.1.3 Detector e Descritor ORB

A definição de *features* ORB envolve a união do detector FAST com o descritor binário BRIEF, e adiciona diversas melhorias. A primeira é a adição de um componente de orientação, tornando o método invariante à rotação. Isso é possível pela adição de uma informação a mais sobre o bloco da *feature*, chamada de *momento da imagem*, um conceito apresentado por (ROSIN, 1999). Nada mais é que uma média ponderada que considera as intensidades dos *pixels*, exemplificada pela Equação 3.2. u e v são todos os *pixels* dentro do bloco e p e q representam os pesos, que assumem os valores $(p, q) = (0, 0)$ ou $(1, 0)$ ou $(0, 1)$.

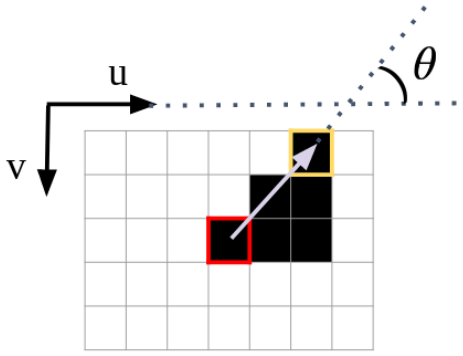
$$m_{pq} = \sum_{u,v} u^p v^q I(u, v) \tag{3.2}$$

Em seguida, pela Equação 3.3, é calculado o pixel do bloco que representa o *centro de massa*.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3.3}$$

A Figura 28 ilustra uma orientação θ de um bloco. Os *pixels* com borda em amarelo e vermelho (que indicam o centro de massa e centro do bloco, respectivamente) formam um vetor. θ é, portanto, o ângulo definido entre esse vetor e o eixo u do sistema de coordenadas da imagem.

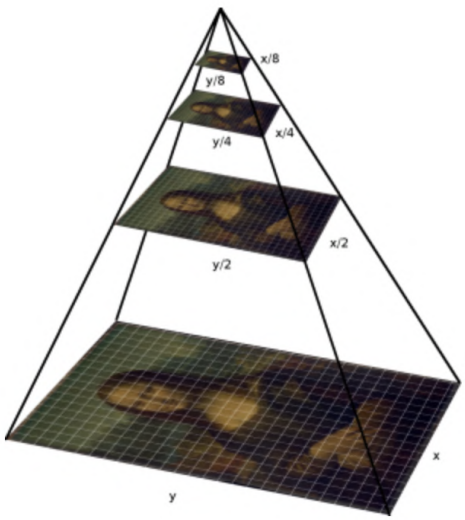
Figura 28 – Orientação θ do bloco.



Fonte: autora.

Além de invariante à rotação, ORB é invariante à escala. Para isso, uma pirâmide da mesma imagem com diferentes resoluções é adotada, como mostrada na Figura 29. Cada nível contém uma resolução menor do que a imagem do nível anterior e o algoritmo FAST é utilizado para detectar *features* em cada uma dessas imagens.

Figura 29 – Cada nível da pirâmide contém uma resolução menor do que a imagem do nível anterior.



3.3.2 Correspondência de *Features*

A correspondência de *features* ocorre de três formas diferentes, podendo ser uma correlação 2D-2D, 3D-2D ou 3D-3D. Sabe-se que uma *feature* possui um descritor representado por um vetor binário (ver Seção 3.3.1), e o ponto 3D, que é proveniente de uma *feature*, é representado pelo mesmo vetor descritor. Ou seja, independente da forma que se deseja comparar, o meio é o mesmo: através da distância de Hamming. A distância de Hamming retorna a indicação de quantos *bits* entre dois vetores são diferentes, o que significa que quanto menor a pontuação, mais próxima é a correspondência.

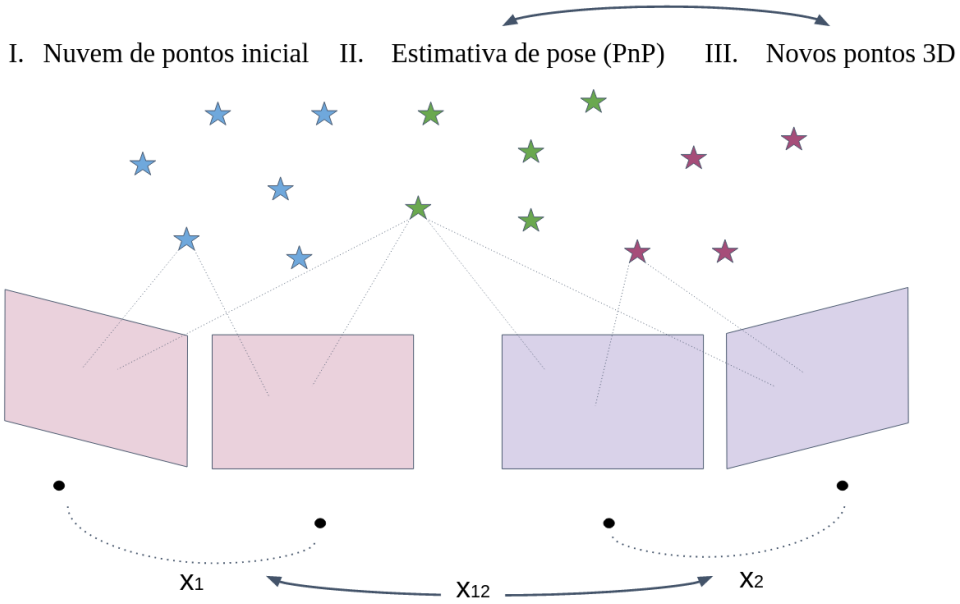
3.3.3 Estimativa da Pose e Triangulação

A associação 3D-2D de *features* é utilizada para uma primeira estimativa da pose (sem otimização). Dado um quadro de imagem, os descritores das *features* extraídas são comparadas com todos os descritores dos pontos 3D do mapa local. Uma vez que as correspondências são definidas, o método (em inglês, Perspective-n-Point) (LEPETIT; MORENO-NOGUER; FUA, 2009) busca estimar a pose de uma câmera dado um conjunto de pontos 3D no mundo e suas projeções 2D correspondentes na imagem. Essa, por exemplo, é a mesma técnica adotada para estimar os parâmetros da câmera, pois é sabido o modelo espacial e deseja-se estimar a pose com base nisso. É importante ressaltar que o método RANSAC (em inglês, RANdom SAmple Consensus) é também adotado a fim de remover correspondências que representem *outliers*.

A correspondência de *features* entre duas imagens do par estéreo é utilizada para inserir novos pontos 3D ao mapa esparso, e só é feita com *keyframes*. *Features* da imagem esquerda e da direita são extraídas e comparadas através de seus descritores. Como é sabido os parâmetros da câmera e a posição atual da mesma, o método de triangulação, já abordado na Seção 2.2, é executado, e os novos pontos são adicionados ao mapa.

Até aqui, o processo ocorre como ilustrado na Figura 30 e descrito da seguinte forma: (a) para a inicialização, a primeira estimativa de pose x_1 gera a nuvem de pontos inicial, representada pelas *features* em azul e verde; (b) as *features* em verdes também são visualizadas pelo próximo par de imagens, sendo estas utilizadas para a estimativa de pose x_2; (c) novos pontos 3D (*features* em rosa) são formados pelo par de imagens atual x_2. As etapas (b) e (c) seguem em repetição. O refinamento dessas estimativas são abordadas na Seção 3.4.

Figura 30 – *Features* são representadas por estrelas. As *features* em azul e verde são visualizadas pelo par de imagens em x_1. Verdes e rosas são vistas pelo par de imagens em x_2.



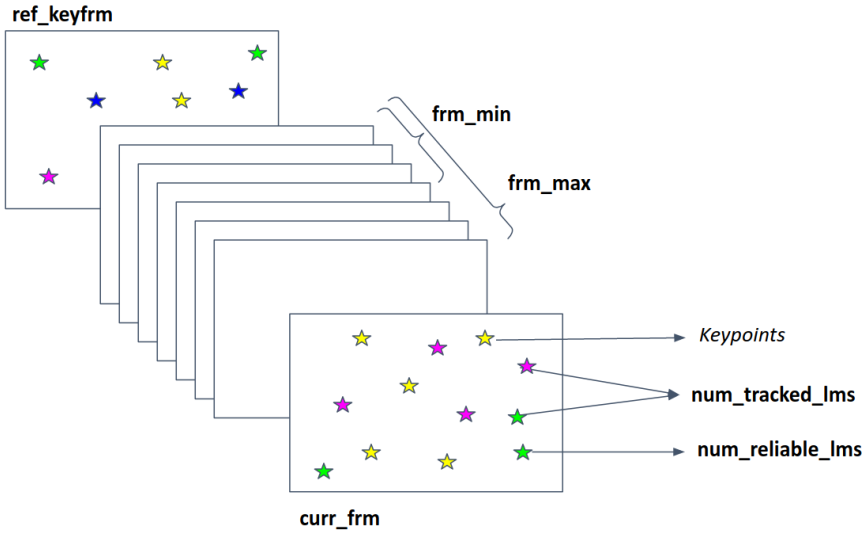
Fonte: autora.

3.3.4 Definição do *Keyframe*

Sequências de imagens são inseridas no sistema SLAM, usualmente, com uma taxa de 15-30 FPS (quadros por segundo). O algoritmo além de não conseguir processar nessa frequência (localização, mapeamento e verificação de revisita), guardar todos os dados representaria alto consumo de memória. No entanto, muitas imagens são redundantes (ou impróprias) e podem ser ignoradas para a execução dos módulos de mapeamento e otimização global (ver Figura 25).

Enquanto a estimativa de localização é feita para todos os quadros de imagem, demais módulos do OpenVSLAM só são executados para os chamados *keyframes*. Sua criação segue uma série de critérios. A Figura 31 ilustra as variáveis que são consideradas para adicionar, ou não, um novo *keyframe* no banco de dados. São eles: $curr\_frm$ indica o quadro atual em análise; ref_keyfrm indica o último *keyframe* adicionado no banco de dados; frm_min e frm_max representam, respectivamente, o valor mínimo e máximo de quadros que se passaram até o momento do quadro atual em análise; as *features* são divididas entre $num\_tracked\_lms$ e $num\_reliable\_lms$, onde o primeiro caso indica o número de *features* que tem representação 3D (já foi mapeada na nuvem) e o segundo significa o número de *features* que tem representação 3D e que também é visualizado por determinado número de *keyframes*.

Figura 31 – Ilustração das variáveis levadas em consideração para adicionar um novo *keyframe*.



Fonte: autora.

- Condição A1: essa verificação garante um espaçamento máximo entre o último *keyframe* e o quadro atual, isto é, retorna verdadeiro se, e somente se, o número de quadros passados após a inserção do último *keyframe* atingir determinado limite ($curr\_frm.id \geq frm\_max$);
- Condição A2: essa verificação retorna verdadeiro se, e somente se, módulo de mapeamento do sistema SLAM estiver em espera (ocioso), e o número de quadros passados após a inserção do último *keyframe* exceder determinado valor mínimo ($curr\_frm.id > frm\_min$ e map_is_idle).
- Condição A3: retorna valor verdadeiro se, e somente se, campo de visão do quadro atual apresentar uma mudança considerável ($num\_tracked\_lms < 0.25{*}num\_reliable\_lms$);
- Condição B: retorna valor verdadeiro se, e somente se, o número de *features* extraídas exceder um valor limite ($num\_tracked\_lms > num\_tracked\_lms\_thr$).

A condição final é ilustrada na Equação 3.4. É possível observar que a adição de um novo *keyframe* só será verdadeiro se, e somente se, a condição B for satisfeita, juntamente com qualquer uma das condições $A1$, $A2$ ou $A3$.

$$is\_keyframe = cond_B \cap (cond_{A1} \cup cond_{A2} \cup cond_{A3}) \tag{3.4}$$

3.3.5 Revisita

Em um sistema vSLAM, um dos desafios é conhecido por *reconhecimento de lugar* (em inglês, *place recognition*). Três maneiras de fazer isso é por meio da comparação de mapa

com mapa, de imagem com imagem ou de imagem com mapa. A opção de avaliação entre imagens é a adotada no sistema OpenVSLAM, sendo esta a mais adequada para mapeamento de cenas de larga escala que as demais. Além disso, as outras técnicas são mais desafiadoras para mapas esparsos (WILLIAMS et al., 2009). Uma visão geral sobre *reconhecimento de lugar* pode ser encontrada no trabalho descrito em (LOWRY et al., 2015).

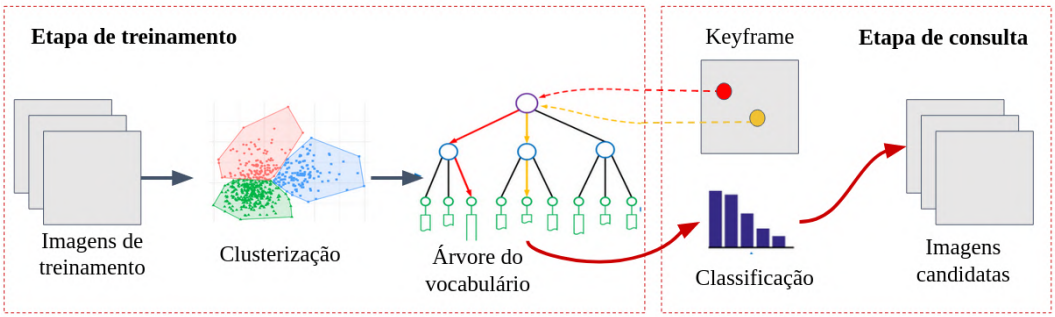
Para detectar a revisita dado uma imagem, é necessário buscar na base de dados imagens com alta probabilidade de correspondência. A técnica utilizada é a Bags of Visual Words (BoW) - *saco de palavras* (*bags of visual words*, em inglês), inicialmente apresentada em (NISTER; STEWENIUS, 2006), e aprimorada para alcançar mais rapidez com o uso de "palavras binárias" - BDoW2 (GÁLVEZ-LÓPEZ; TARDOS, 2012).

A Figura 32 mostra a etapa de treinamento e de consulta da técnica de BoW . Na primeira, ocorre um treinamento *offline* a partir de uma base de dados de imagens. Dessas imagens são extraídas as *features*, as quais são alocadas em um agrupamento hierárquico *k-means* (LLOYD, 1982). Em seguida, a estrutura chamada de árvore de vocabulário (NISTER; STEWENIUS, 2006) é utilizada para acelerar a indexação de similaridade, utilizada como um categorizador visual. Em comparação com os métodos tradicionais baseados em busca exaustiva, os métodos baseados em árvore hierárquica melhoram significativamente o desempenho (LI et al., 2018). A técnica DBoW2 (GÁLVEZ-LÓPEZ; TARDOS, 2012) realiza o agrupamento a partir da similaridade das *features* e de imagens capturadas numa determinada janela de tempo, computando uma pontuação para o grupo, que é a soma das pontuações individuais.

Na etapa de consulta, mostrada na Figura 32, é extraído do quadro de imagem em questão (no nosso caso, um *keyframe*) as *features*, as quais são atribuídas a uma palavra visual uma vez que atravessam a árvore de vocabulário computada na etapa de treinamento. A saída é um vetor de saco de palavras, contendo uma pontuação referente à frequência do termo para cada palavra contida na imagem. Quanto mais presente for a palavra na imagem e menos presente no conjunto de dados de treinamento, maior é esta pontuação. Imagens candidatas no grupo com a pontuação mais alta são selecionadas e a imagem com a pontuação individual máxima é considerada como candidata a revisita da imagem de consulta.

Caso o rastreamento tenha falhado, por exemplo, devido à movimentação brusca da câmera, a busca por reconhecimento do lugar é executada, e uma vez que é encontrado o melhor *keyframe* candidato, a pose da câmera é recuperada usando o algoritmo (LEPETIT; MORENO-NOGUER; FUA, 2009), assim, o sistema SLAM volta à atuar. No caso de ser um *keyframe* que represente uma revisita, o sistema realiza então a etapa de otimização do grafo, detalhada na Seção 3.4.

Figura 32 – A fase de treinamento constrói, baseado em um conjunto de imagens, uma estruturação de árvore hierárquica; a fase de consulta retorna imagens semelhantes ao *keyframe* em análise.



Fonte: autora.

3.4 OTIMIZAÇÃO DO GRAFO: BACK-END

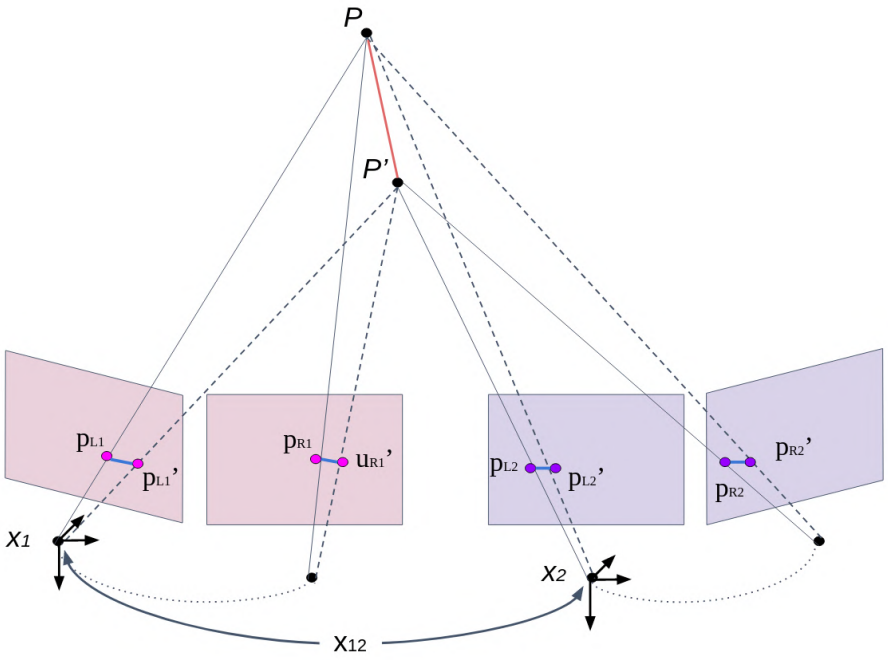
Num sistema SLAM, o mapeamento e a estimativa de localização da câmera são ambos desconhecidos. Dessa forma, quanto mais se mapeia e mais se estima a localização, mais se acumula erro. As etapas de *Bundle Adjustment* (BA e *otimização do grafo de poses* são técnicas utilizadas pelo SLAM para mitigar o erro das estimativas. Ambas representam otimizações não-lineares, as quais podem ser resolvidas através do método Gauss-Newton padrão, e de variantes do algoritmo Levenberg-Marquardt (KÜMMERLE et al., 2011). A diferença entre eles é que o BA considera na solução nós do grafo que representam pontos 3D e *keyframes*, enquanto que a otimização do grafo de poses, apenas nós que indicam *keyframes* são levados em consideração.

3.4.1 Bundle Adjustment

Bundle Adjustment é um processo de otimização que busca o ajuste das poses estimadas e dos pontos 3D do mapa a fim de minimizar o erro entre esses dois elementos. Isto é possível através da re-projeção dos pontos nos quadros de imagem, como ilustrado na Figura 33, com apenas um único ponto 3D e dois pares de imagens. Num primeiro momento, o par de imagens indicado por rosa cria o ponto P no espaço. Os planos de imagens em roxo indicam um novo par de imagens que também enxerga o ponto P, porém, baseado na estimativa de pose x_{12} (pose de 1 em relação a 2, a representação tridimensional de P se dá em P'. A técnica BA vai re-projetar esse ponto P' nos planos de imagem em rosa, representados por p'_{L1} e p'_{L2} e o ponto P nos planos de imagem em roxo, indicados por p'_{L2} e p'_{L2}. As distâncias entre projeções e entre pontos 3D equivalem ao erro o qual se deseja minimizar, isto é, $p - p'$ e $P - P'$.

Partindo do pressuposto que são conhecidos os parâmetros intrínsecos e extrínsecos do sistema estéreo, representados pela matrizes K e M, respectivamente; e sendo n o número de pontos 3D e m o números de poses x que se deseja ajustar, a Equação 3.5 representa o vetor de variáveis a serem ajustadas e sua respectiva quantidade de dados, considerando que

Figura 33 – Re-projeção dos pontos 3D do mapa nos planos de imagem. O erro é definido por $p - p'$ ou $P - P'$, onde a técnica de BA busca otimizar para todas as poses e pontos 3D em questão.



Fonte: autora.

a pose tem seis graus de liberdade e o ponto 3D, três graus de liberdade.

$$\{x_1, x_2 ... x_m | P_1, P_2 ... P_n\} \in \mathbb{R}^{6m+3n} \tag{3.5}$$

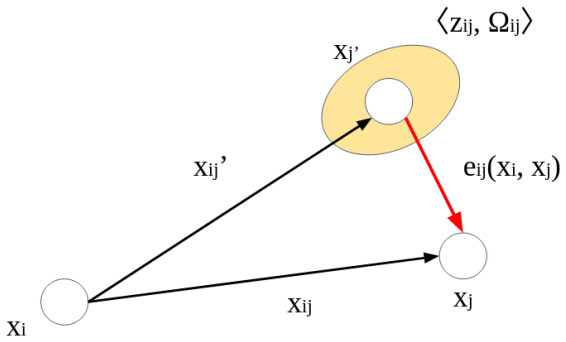
É importante ressaltar que a estimativa na pose é usualmente referida apenas para a câmera esquerda, mas como *baseline* do sistema é fixo, pode-se facilmente calcular a pose para a câmera direita. Dessa forma, para cada par de imagens há apenas uma estimativa de pose, por isso, no exemplo da Figura 33, apesar de existir quatro quadros de imagem, o ajuste da técnica BA aconteceria como se apenas houvessem dois quadros, reduzindo o número de variáveis da Equação 3.5.

A utilização do BA no sistema SLAM pode se dar de duas formas: localmente ou globalmente. O BA local otimiza o *keyframe* atual e demais *keyframes* conectados a ele no grafo, além de todos os pontos 3D do mapa vistos por eles. Outros *keyframes* (que não estão conectados no grafo com o *keyframe* em questão) que visualizam esses mesmos pontos 3D são considerados no cálculo do BA local, porém permanecem fixos. No método BA global, todos os *keyframes* e pontos no mapa são otimizados, exceto o *keyframe* de origem que é fixo. A técnica BA pode ser formulada como um problema de otimização de mínimos quadrados, onde a implementação do algoritmo Levenberg-Marquardt (KÜMMERLE et al., 2011) é utilizada.

3.4.2 Otimização do Grafo de Poses

Uma chance de diminuir o acúmulo de erro está no momento em que uma revisita é detectada (A técnica para reconhecer a revisita foi abordada na Seção 3.3.5). Isso porque duas informações podem ser usadas para calcular o erro: a pose estimada de acordo com o grafo e o dado atual do sensor. A Figura 34 ilustra três posições. Nesse exemplo, x_i é a pose estimada pelo sistema SLAM antes da detecção de revisita; x_j é a estimativa da pose no momento que a revisita foi detectada e ela leva em conta todo o acúmulo de erro durando a criação do grafo; x'_j é a posição que x_j *deveria* estar se considerássemos somente a leitura do sensor daquele momento, ou seja, considerando apenas os pontos 3D visíveis pela imagem, representado por z_{ij}; Ω_{ij} é a matriz de informação que indica a região de incerteza (elipse em amarelo).

Figura 34 – Ilustração de três posições: a pose estimada antes (x_i) e depois (x_j) da detecção de revisita, e a posição que x_j estaria (x'_j) pela leitura do dados z_{ij}; Ω_{ij} indica a região de incerteza.



Fonte: autora.

A medida do erro e_{ij}, destacada em vermelho na Figura 34, é formalizada pela Equação 3.6, onde o termo $(X_i^{-1}X_j)$ relaciona a pose x_i com respeito à pose x_j e o termo Z_{ij}^{-1} representa as medidas z_{ij} com respeito à x_j. Fica evidente, portanto, que toda a operação é uma transformação matricial, sendo que o erro deve ser representado em formatação vetorial, justificando o termo $t2v$ - "transformada para vetor". É importante ressaltar que o erro vai sempre relacionar apenas duas poses por vez, as quais estão ligadas por uma aresta no grafo.

$$e_{ij}(x) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \tag{3.6}$$

A otimização de poses, portanto, tenta deslocar e ajustar todas as poses do grafo de forma que os erros sejam minimizados, como mostrado na Equação 3.7, onde x representa o grafo de poses e seus respectivos valores estimados, e F_k representa o custo associado à uma aresta k que conecta duas posições x_i e x_j.

$$x^* = \underset{x}{argmin} \sum_k F_k(x) \tag{3.7}$$

O custo, por sua vez, é formalizado pela Equação 3.8. Nela, é possível observar que quando maior a região de incerteza Ω_{ij}, maior o custo associado.

$$F_k(x) = \underset{x}{argmin} \sum_k e_k^T(x)\Omega_k e_k(x) \tag{3.8}$$

Depois da otimização do grafo de poses, é realizada a técnica de BA globalmente a fim de obter a solução ideal do grafo, que agora convergirá mais rápido, dado que a saída de otimização do grafo de pose já está mais próxima da ideal.

4 TRABALHOS RELACIONADOS

Sistemas vSLAM representam o ambiente mapeado através de uma nuvem de pontos, seja ela esparsa, semi-densa ou densa. Para transformar a saída desses sistemas em mapas adequados à navegação (discutido na Seção 1.1), um *framework* de mapeamento, como o OctoMap (HORNUNG et al., 2013), é uma alternativa. Este é capaz de converter qualquer nuvem de pontos, independente do sensor utilizado, para um mapa de ocupação 2D ou 3D (OctoMap). O sucesso da conversão, no entanto, requer no mínimo uma nuvem de pontos semi-densa, sendo inviável para a técnica proposta, na qual o vSLAM produz uma representação esparsa da cena. O OctoMap é um projeto de código aberto, e sua proposta funciona como um pós-processamento, onde o dado inserido é resultado do mapeamento, isto é, a nuvem otimizada e finalizada. A técnica é, então, aprimorada em (DUBERG; JENSFELT, 2020) para que, junto ao mapeamento, ocorra a atualização e otimização do OGM.

Além do *framework* de mapeamento mencionado, há também outros trabalhos com o mesmo propósito de alcançar um mapa mais adequado para navegação, como é o caso da nossa proposta. Dessa forma, a revisão bibliográfica teve como foco pesquisas que se desenvolveram em cima de um sistema vSLAM, seja ele baseado no método direto ou indireto. É importante pontuar que os recentes projetos de código aberto exemplificados pelo ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) e LSD-SLAM (ENGEL; SCHÖPS; CREMERS, 2014), adquiriram grande adesão de contribuidores e representam importantes marcos. Estes foram de grande valia para essa linha de pesquisa, onde apenas dois dos trabalhos discutidos a seguir não usou um projeto de código aberto em seu desenvolvimento.

A Tabela 2, resume os trabalhos mais recentes encontrados. Para cada referência, está destacado: o sistema vSLAM e seu respectivo método (direto ou indireto); os sensores envolvidos; o tipo de mapa final resultante e utilizado para navegação; e a informação sobre o funcionamento da proposta, isto é, se consiste num pós-processamento ou se roda junto com o algoritmo SLAM em tempo real (RT). As cores destacadas na Tabela 2 separam os trabalhos que constroem os mapas à partir da nuvem densa dos que usam diretamente a nuvem esparsa.

Tabela 2 – Trabalhos que geram a partir de um sistema vSLAM um mapa apropriado para navegação. As cores destacadas separam os trabalhos que constroem os mapas à partir da nuvem densa dos que usam diretamente a nuvem esparsa.

Referência	SLAM	Método	Sensor	Mapa	RT?
(YANG et al., 2017)	ORB-SLAM2	Indireto	RGB-D	OGM 3D	Sim
(YAN; LAN; YANG, 2020)	VINS-Mono	Indireta	Monocular + IMU	OGM 3D	Não
(LING; SHEN, 2019)	CBSLAM	Indireto	Estéreo	TSDF	Sim
(XU et al., 2019)	ORB-SLAM2	Indireto	RGB-D	OGM 2D	Sim
(LABBÉ; MICHAUD, 2019)	RTAB-Map	Indireto	RGB-D/laser/estéreo	OGM 2D/3D	Sim
(STUMBERG et al., 2016)	LSD-SLAM	Direto	Monocular	OGM 3D	Sim
(ESRAFILIAN; TAGHIRAD, 2016)	ORB-SLAM2	Indireto	Monocular	OGM 3D	Não
(LING; SHEN, 2017)	ORB-SLAM2	Indireto	Estéreo	Triâng. de Delaunay	Sim
(CHEN; LIU, 2021)	ORB-SLAM2	Indireto	Estéreo	Regiões convexas	Não
(BLOCHLIGER et al., 2018)	próprio	Indireto	Monocular + IMU	Topológico e TSDF	Não
Proposto (Segfloor)	**OpenVSLAM**	**Indireto**	**Estéreo**	**OGM 2D**	**Sim**

Fonte: autora.

4.1 MAPAS A PARTIR DA NUVEM DENSA

Os trabalhos destacados em cinza escuro da Tabela 2, obtêm seus mapas a partir de duas etapas. A primeira consiste na estimativa de posição do agente junto à uma representação esparsa do ambiente, onde ambos são definidos por um sistema vSLAM baseado em *features*. A segunda etapa é a geração da representação volumétrica/densa do ambiente, cuja projeção dos pontos no mapa 3D toma como base o mapa global gerado na primeira etapa.

O trabalho proposto em (YANG et al., 2017) utiliza o ORB-SLAM2 apenas como um módulo de localização, e com relação ao mapeamento, adiciona a nuvem de pontos densa formada pelo sensor RGB-D. A medida que o sistema SLAM otimiza as estimativas, a correção é replicada para a nuvem de pontos densa. Para gerar o mapa OGM 3D, os autores associam a técnica com o OctoMap, no entanto, adicionam melhorias para (a) funcionar online, isto é, junto ao algoritmo SLAM e (b) para apresentar uma estruturação dos dados em árvore que leva em consideração o número de identificação do *keyframe*, sendo uma proposta mais eficiente quando comparada ao projeto original.

O tempo de processamento para manter a estrutura de dados cresce linearmente com o número de nós da árvore. Pelos resultados apresentados, é possível observar que para cada novo *keyframe* inserido há um acréscimo de 1 milésimo de segundo para processar a estrutura. Dessa forma, para uma cena com 500 *keyframes*, o custo seria de meio segundo, comprometendo seu funcionamento em tempo real. Para uma cena de larga escala, como a suportada pelo método proposto, com mais de mil *keyframes*, essa técnica levaria mais de 1 segundo para manter seu módulo de mapeamento.

Na proposta apresentada em (YAN; LAN; YANG, 2020) é utilizado o sistema SLAM baseado em *features* nomeado de VINS-Mono. Este mapeia e se localiza no ambiente a partir de uma câmera monocular e de um sensor inercial. O quadro de imagem vai tanto para o módulo

SLAM quanto para o módulo de estimativa de profundidade, a qual é feita de forma densa através de redes neurais profundas previamente treinadas. A técnica apresentada em (CHEN et al., 2018), que é originalmente usada para segmentação semântica, foi adaptada para estimar a profundidade dado uma única imagem.

Enquanto o algoritmo SLAM fornece a posição, o módulo de profundidade gera a nuvem de pontos densa, projetando os pontos da imagem no espaço 3D baseado na respectiva posição com relação ao sistema de coordenadas global. Esse processo é realizado apenas para cada *keyframe*, onde potenciais *outliers* são removidos considerando a média da distância entre os pontos 3D vizinhos, isto é, a densidade local. A nuvem de pontos global é então utilizada para gerar o mapa OGM 3D através de uma adaptação do projeto OctoMap que se mostra mais eficiente que o trabalho original (YANG et al., 2017).

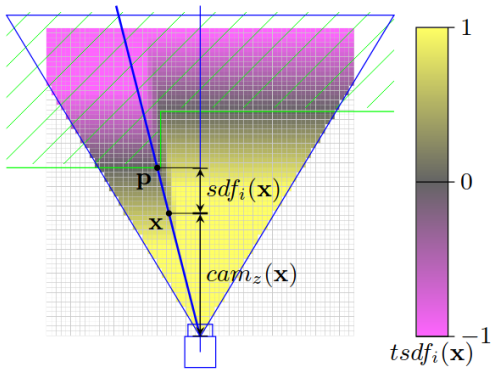
A avaliação considera uma cena de pequena escala, do tamanho de um cômodo. Foram gerados 75 *keyframes*, e para este caso, o processo levou 15.0 segundos estimando a profundidade e 9.9 segundos gerando a nuvem de pontos densa. Mesmo o experimento sendo realizado com auxílio de uma GPU, o tempo é grande o bastante para impedir que a técnica proposta seja usada em tempo real.

O trabalho descrito em (LING; SHEN, 2019) é majoritariamente focado na proposta de um sistema SLAM baseado em *features* com mapeamento esparso, nomeado CBSLAM (em inglês, *complexity-bounded SLAM*), o qual é posto em comparação com o ORB-SLAM2. Mas além do conceito do algoritmo SLAM em si, é acrescido também um mapeamento denso voltado para o planejamento de rotas, cuja estimativa de profundidade de todos os *pixels* da imagem é realizada usando o algoritmo de disparidade SGM (HIRSCHMULLER, 2008), ou ELAS (GEIGER; ROSER; URTASUN, 2010), ou uma abordagem própria do trabalho.

O ambiente é modelado em *voxels*, onde cada um é associado a uma função de distância sinalizada truncada - TSFD (em inglês, *truncated signed distance function*) (CURLESS; LEVOY, 1996; IZADI et al., 2011). Na representação Signed Distance Function (SDF) (em inglês, *truncated signed distance*), a qual indica um caso que "antecede" a Truncated Signed Distance Function (TSDF) (OLEYNIKOVA et al., 2016), cada *voxel* é relacionado com dois valores relevantes. O primeiro é determinado pela função $sdf_i(x)$, que é a distância entre o centro do *voxel* e a superfície do objeto mais próximo considerando a direção do feixe do sensor, como destacado na Figura 35, onde o subscrito i denota a i-ésima observação. Na frente de um objeto (no espaço livre), os valores são definidos como positivos. Atrás da superfície ("dentro" do objeto) as distâncias são negativas. O segundo valor é associado à incerteza de $sdf_i(x)$. A variação da SDF, a TSDF é uma função $tsdf_i(x)$ utilizada para restringir os valores a serem considerados, isso porque grandes distâncias podem não ser relevantes. Vale salientar que essa variação favorece um menor consumo de memória. Além do mais, o trabalho possui execução de seis módulos em paralelo, executados de forma assíncrona.

Ainda sobre o mapeamento denso apresentado em (LING; SHEN, 2019), os autores dividem os *voxels* em grupos, os quais são associados à estimativa de pose dos *keyframes*. Pela Figura

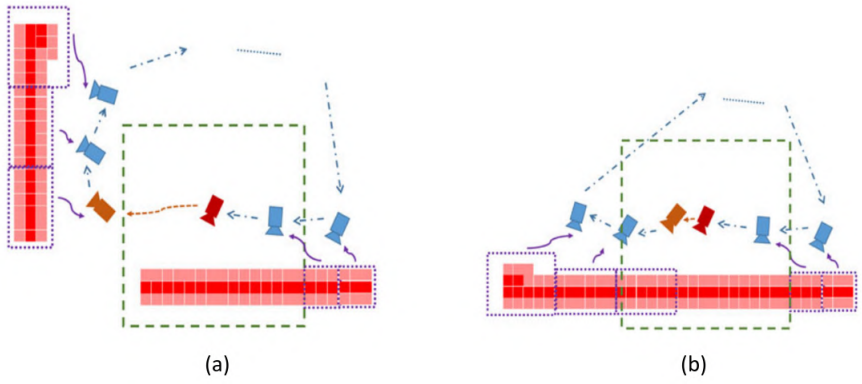
Figura 35 – Representação de um ambiente modelado por *voxels*, onde são destacadas as funções $sdf_i(x)$ e $tsdf_i(x)$, utilizadas para adicionar novos dados aos *voxels*. x representa o valor da distância sinalizada, determinado em função da profundidade do ponto p e da distância da câmera $cam_z(x)$.



Fonte: (WERNER; AL-HAMADI; WERNER, 2014).

36 essa associação é indicada pelas setas roxas, em que é ilustrada também a efetividade dessa abordagem para a correção das estimativas. A Figura 36 mostra (a) mostra o momento antes da otimização e (b) o resultado da mesma, onde os grupos de *voxels* são movimentados em conjunto com o *keyframe* relacionado. Vale salientar que essa reconstrução densa é feita considerando o mapa local, portanto, sua complexidade não varia em relação à distância percorrida.

Figura 36 – Modelagem do ambiente em grupos de *voxels*, os quais são associados com determinada posição da câmera (setas roxas). A utilização da técnica é ilustrada (a) antes e (b) depois da otimização do mapa.



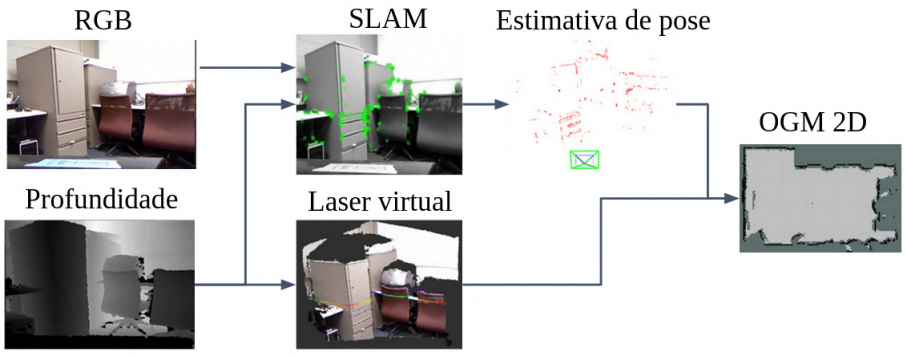
Fonte: (LING; SHEN, 2019).

Testes foram realizados a partir de base de dados públicas, em que um deles resulta em mais de 5 mil *keyframes*. Apesar do ORB-SLAM2 apresentar melhores resultados com relação à estimativa da posição da câmera, os autores defendem a busca de um equilíbrio entre a precisão na localização e a densidade do mapeamento. Para a base de dados com mais de 5 mil *keyframes*, o consumo de memória é tanto que o ORB-SLAM2 não suporta mais do que

24 minutos de vídeo, o que não acontece com o CBSLAM devido a sua estratégia de memória apresentada, exemplificada pela remoção de *keyframes* redundantes, de pontos 3D desnecessários e de *outliers*. A abordagem funciona em tempo real tanto em GPU quanto CPU. Além do mais, pode ser associada à um sensor inercial, o qual melhora consideravelmente a estimativa da localização. Testes foram extensivamente realizados, os quais contemplam ambientes *outdoors*. No mais, não foram realizados experimentos sobre a atividade de navegação em si.

Na técnica apresentada em (XU et al., 2019), os autores desenvolveram um mapa grade de ocupação a partir do sistema ORB-SLAM2, fazendo uso do sensor RGB-D. A Figura 37 ilustra como isso foi possível. Inicialmente, a imagem RGB e a informação de profundidade entram no sistema SLAM, responsável por estimar a posição da câmera. Por outro lado, apenas a informação de profundidade é usada como entrada para um módulo de laser *virtual*. Nele, é extraído a partir da nuvem densa os pontos que estão à uma determinada altura, simulando um laser 2D. Essa "fatia" horizontal está destacada do módulo *laser virtual* da Figura 37. Além dessa proposta representar um elo entre um sistema SLAM de mapeamento esparso e atividades de navegação, os autores acrescentam que essa associação representa uma ferramenta mais intuitiva e interativa para o usuário.

Figura 37 – Fluxo de desenvolvimento do trabalho (XU et al., 2019) para gerar o mapa OGM 2D.



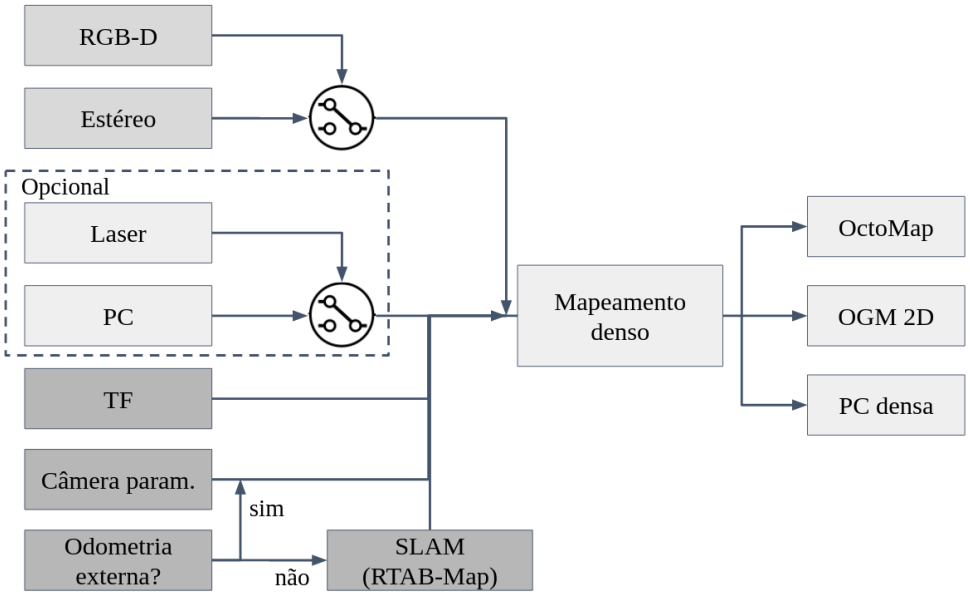
Adaptada: (XU et al., 2019).

Durante o mapeamento, é realizada a associação entre os *keyframes*, e as suas respectivas estimativas de pose com os pontos provenientes do laser virtual. Dessa forma, quando o sistema SLAM fizer a correção do acúmulo de erro, o mapa OGM também pode ser corrigido. Ao fim do mapeamento, toda a associação e o próprio mapa 2D é serializado junto à base de dados do ORB-SLAM2. Dessa forma, quando o agente utilizar o sistema SLAM apenas para se localizar, o OGM também é carregado, e a posição 3D da câmera estimada, é convertida e projetada para o mapa 2D. Vale salientar que os resultados apresentados são relativos à uma sala de escritório e um corredor *indoor*. A avaliação é voltada para a acurácia apenas da localização do sistema SLAM, onde custos computacionais não são reportados.

O RTAB-Map (LABBÉ; MICHAUD, 2019) é um projeto de código aberto. Além de possuir

um módulo independente de um sistema SLAM baseado em *features*, ele também apresenta, de forma desacoplada, um módulo de mapeamento para que, assim, alcance maior densidade de mapa. Baseado em como o usuário configura seu funcionamento, o sistema pode funcionar com apenas um dos módulos, ou os dois em série. Isso quer dizer que até mesmo outras abordagens que geram a informação de odometria podem ser integradas ao RTAB-Map. A Figura 38 mostra as entradas necessárias para o funcionamento da proposta. O sensor de imagem pode ser uma câmera RGB-D ou câmera estéreo. De forma opcional, o usuário pode associar um desses sensores com a informação de laser ou nuvem de pontos. Neste método, são também requeridos os parâmetros de calibração da câmera, a configuração de sistemas de coordenadas do agente robótico, por exemplo, a relação entre as coordenadas de câmera e a base do robô, e a informação de odometria externa. Caso não haja odometria externa, o algoritmo SLAM do próprio RTAB-Map estima o dado.

Figura 38 – Entradas requeridas pelo trabalho (LABBÉ; MICHAUD, 2019) e as respectivas saídas com relação ao mapeamento. Adaptada: (LABBÉ; MICHAUD, 2019).

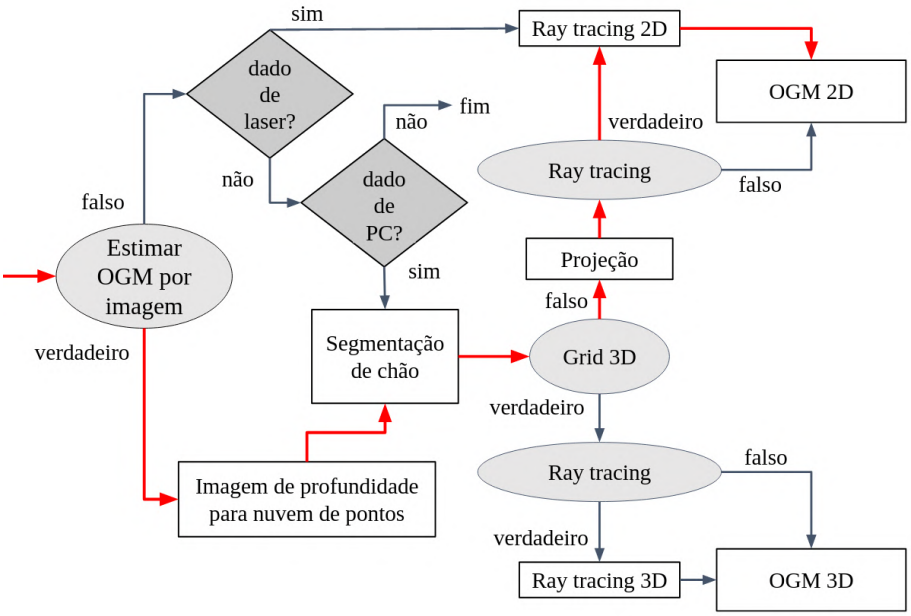


Adaptada: (LABBÉ; MICHAUD, 2019).

Na medida que os dados requeridos entram no sistema, é gerado um mapeamento denso para que, a partir dessa nuvem de pontos, se alcance o OGM 2D e/ou 3D (OctoMap). A forma como esses mapas são formados é detalhada a seguir. Seu maior detalhamento é importante uma vez que comparamos a técnica proposta (nomeada de *Segfloor*) com essa abordagem. A Figura 39 mostra as diversas possibilidades de fluxo do processamento para gerar o OGM final, o qual é determinado por um arquivo de configuração. O fluxo adotado neste trabalho é destacado pelas setas em vermelho e detalhado a seguir.

Quando as imagens estéreo, junto com o dado de odometria externa proveniente (no nosso caso dado pelo sistema OpenvSLAM (SUMIKURA; SHIBUYA; SAKURADA, 2019)) entram

Figura 39 – Fluxograma do trabalho (LABBÉ; MICHAUD, 2019). O arquivo de configuração é responsável por definir qual sentido o fluxo irá percorrer. As setas vermelhas indicam a configuração adotada no nosso trabalho.



Adaptada: (LABBÉ; MICHAUD, 2019).

no módulo de mapeamento do RTAB-MAP, este gera o mapa de disparidade densa, seguido da estimativa de profundidade, resultando, assim, a nuvem de pontos densa relativa às imagens da iteração atual. Toda a área ocupada pela nuvem de pontos é dividida em *voxels* (RUSU; COUSINS, 2011). A partir daí, eles são classificados como *chão* ou *não-chão* pela técnica conhecida como *plane fitting*, ou seja, é calculado o plano que melhor inclui os respectivos pontos do *voxel* em questão. O vetor normal paralelo (ou dentro de uma tolerância) ao eixo z são rotulados como *chão*, e os demais como *obstáculos*. Se a opção de *grade 3D* estiver definida como falsa, o resultado da classificação dos *voxels* são projetados em grades 2D, prevalecendo a classificação de obstáculos durante a mesclagem. Em seguida, de forma opcional, é realizado o tracejado de raio (em inglês, *ray tracing*), que consiste em um feixe de raios que vão da posição atual da câmera até todos os *voxels*/grades ocupados visíveis no momento. Os *voxels*/grades atravessados por esse feixe são classificados como região livre. De forma incremental, os mapas de ocupação são definidos. Ocorrência de revisita é constantemente verificada a fim de corrigir acúmulo de erro da técnica (LABBE; MICHAUD, 2013).

A abordagem proposta usa um fluxo bastante semelhante ao destacado nas setas em vermelho da Figura 39, mas ao invés de gerar uma nuvem de pontos densa, usamos diretamente a nuvem esparsa produzida pelo sistema SLAM. Essa mudança é responsável por reduzir o tempo de processamento e uso de memória necessários para alcançar um OGM 2D, conforme mostrado no Capítulo 7 de resultados.

Os trabalhos abordados nessa seção utilizam um sistema vSLAM baseado em *features*,

que é menos custoso, apenas para prover a informação de localização. Em contra-partida, seu mapeamento esparso é substituído por um denso ou semi-denso. Apesar dos recentes avanços (LING; SHEN, 2019), o mapa volumétrico obtido é custoso e nem todas as aplicações de navegação necessitam, ou suportam, um mapeamento 3D, exemplificado por um robô móvel operando com embarcados de recursos limitados, onde um mapa 2D seria suficiente para sua navegação. A proposta apresentada nessa dissertação tem como prioridade o baixo consumo de memória e tempo de processamento, e dispensa a etapa de mapeamento denso da nuvem de pontos, buscando produzir o OGM 2D diretamente da nuvem esparsa, permitindo um processo que funciona em conjunto com o sistema SLAM em tempo real.

4.2 MAPAS A PARTIR DA NUVEM ESPARSA OU SEMI-DENSA

Nesta seção são abordados os trabalhos destacados em cinza claro da Tabela 2, os quais obtêm seus mapas para navegação diretamente da nuvem esparsa, com exceção do trabalho (STUMBERG et al., 2016) que já parte da representação semi-densa do ambiente. São propostas que realizam um enriquecimento de densidade a partir da informação esparsa do sistema SLAM utilizado. É neste princípio que a nossa abordagem também é caracterizada.

O trabalho de Stumberg (STUMBERG et al., 2016) propõe o alcance do OGM 3D a partir de um drone equipado com uma câmera monocular. Inicialmente, é obtida uma nuvem de pontos semi-densa através do sistema SLAM de método direto, o LSD-SLAM (ENGEL; SCHÖPS; CREMERS, 2014), que determina a profundidade dos *pixels* de alto gradiente. O OctoMap (HORNUNG et al., 2013) também é integrado, o qual recebe a nuvem semi-densa e divide a área mapeada em *voxels*, de forma que a presença de pontos 3D indica os volumes ocupados, e os livres são definidos através de *ray tracing*[1].

Novos pontos são inseridos ao algoritmo do OctoMap toda vez que o sistema SLAM estabelece um novo *keyframe*. O acúmulo de erro, percebido e corrigido pelo LSD-SLAM, é replicado no OGM de forma periódica, onde a técnica proposta recria todo o mapa do zero, o que pode levar alguns segundos. Assim, o drone espera pairando até prosseguir com a exploração, expondo uma técnica de alto custo. Além disso, foi observado que muitos volumes eram classificados como *indefinido*. Na tentativa de atribuir o estado livre ou ocupado para esses casos, os autores propõem que o drone voe seguindo um padrão de movimento no formato de uma estrela, ocupando e cobrindo o máximo possível toda a região navegável. A avaliação da técnica proposta, no entanto, é realizada em um cômodo de pequena escala. É possível pontuar também que o uso exclusivo de uma câmera monocular traz a desvantagem de se desconhecer a real escala da cena.

No trabalho apresentado em (ESRAFILIAN; TAGHIRAD, 2016), é proposta uma estratégia de navegação que consiste num método de reconhecimento de obstáculos e prevenção de colisão para um quadricóptero provido de uma câmera monocular. É utilizado o ORB-SLAM

[1] Feixe de raio que vai da posição atual da câmera até todos os *voxels*/grades ocupados visíveis no momento.

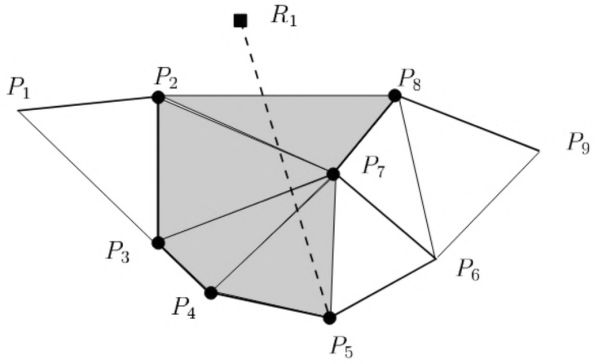
para gerar a nuvem de pontos esparsa do ambiente, a qual é enriquecida para apresentar maior densidade. Para isso, o mapa esparso ao redor do quadricóptero é dividido em *voxels* e apenas aqueles com uma quantidade mínima de pontos são levados em consideração. Para cada *voxel* é então calculado o plano que melhor agrupa os pontos, técnica chamada de *plane fitting*, a mesma utilizada no trabalho (LABBÉ; MICHAUD, 2019). Os planos gerados são categorizados em cinco grupos a depender da direção do vetor normal do plano com relação ao drone, ou seja, apontando para a direita; para a esquerda; de baixo para cima; de cima para baixo; ou em direção ao drone. O *voxel* cujo plano não se enquadra dentro da tolerância angular do respectivo agrupamento é desconsiderado para a próxima etapa, que consiste em uma adição de novos pontos dentro do plano de cada *voxel* remanescente.

Testes foram realizados em um corredor onde obstáculos planares e bem definidos variam de posição e quantidade de uma cena para outra. Todos os dados coletados pelo drone são enviados para um laptop via Wi-Fi para serem processados em tempo real. Depois de mapeado o ambiente, é informado o ponto inicial e o de destino para que o algoritmo RRT (em inglês, *rapidly-exploring random trees*) (LAVALLE et al., 2001) defina o planejamento da rota. Para isso, é calculada, de forma aleatória e incremental, uma árvore de possíveis caminhos até alcançar o ponto de destino. Pelo vídeo disponibilizado (ESRAFILIAN, 2016) é possível perceber considerável instabilidade no mapa e na própria rota. Além do mais, a proposta, claramente, é limitada a objetos planares, evidenciado pelo agrupamento dos planos de cada *voxel*.

No trabalho descrito em (LING; SHEN, 2017), a nuvem de pontos esparsa do sistema SLAM é usada para construir um mapa com fins de navegação. A proposta se utiliza de um sensor estéreo, mas a funcionalidade deve se dar para qualquer entrada que consista numa nuvem de pontos esparsa do ambiente. São três módulos: o primeiro mapeia a cena através do ORB-SLAM2; depois, apenas com a informação de nuvem esparsa, estima uma maior densidade; e em seguida, extrai a região livre de obstáculos. A técnica de triangulação de Delaunay é utilizada para o mapeamento denso. Ela requer como entrada um conjunto de pontos, e a partir disso, forma triângulos de tal maneira que maximiza o ângulo mínimo de todos os ângulos envolvidos. Em seguida, regiões livres de determinados triângulos são definidas a partir das *restrições de visibilidade*. A Figura 40 é utilizada para a compreensão desse método em um cenário 2D. Os pontos $P_1 - P_9$ podem ser interpretados como pontos provenientes do mapeamento esparso do sistema SLAM. R_1 é a posição atual do agente e os pontos destacados com marcadores pretos, por exemplo P_2, indicam os pontos visíveis por ele. A restrição alega que todos os triângulos interceptados pelo raio que sai de R_1 para cada ponto visível, exemplificado por $\overline{R_1 P_5}$, são classificados como região livre (triângulos destacados em cinza).

Essa estratégia, inicialmente abordado em (BUFFA; FAUGERAS; ZHANG, 1992), é a mesma usada no trabalho descrito em (LING; SHEN, 2017), só que em um espaço tridimensional. O cálculo dos triângulos de Delaunay é computacionalmente caro, e trabalhos similares como os abordados em (LABATUT; PONS; KERIVEN, 2007), apresentam uma proposta de pós-processamento,

Figura 40 – Triângulos de Delaunay e a representação de regiões livres (em cinza) pela *restrição de visibilidade*.



Fonte: (BUFFA; FAUGERAS; ZHANG, 1992).

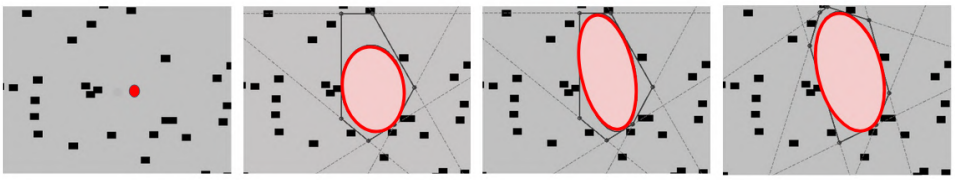
ou em tempo real, mas para uma cena de pequena escala (LOVI et al., 2010). Em contra partida, na abordagem apresentada em (LING; SHEN, 2017) o teste é realizado para um ambiente de larga escala e em conjunto com o sistema SLAM de forma *online*. Isso porque os autores calculam os triângulos considerando o mapa esparso local e o recálculo também local de triângulos, associado à um ponto 3D, só é realizado quando o resultado da otimização do sistema SLAM afeta a posição desse ponto acima de um determinado valor.

Os autores afirmam que, enquanto a maioria dos trabalhos de mapeamento estão preocupados com acurácia, eles focaram em encontrar um compromisso entre acurácia e tempo de processamento, sendo apropriado até mesmo em situações de larga escala. Como a proposta está diretamente relacionada com a nuvem de pontos esparsa, a qual apresenta ruídos, artefatos visuais são observados no vídeo disponibilizado (LING, 2017) durante o mapeamento 3D. Resultados relativos ao custo em função do tempo do mapeamento denso apresenta um comportamento constante devido ao cálculo dos triângulos considerar apenas o mapeamento e a otimização de forma local. No entanto, não foram apresentadas métricas como a média ou desvio padrão desse custo, apenas o gráfico, onde é possível estimar um intervalo de 0.2 à 0.5 segundos, com picos de até 1 segundo na ocorrência de otimização global (detecção de revisita). Por fim, é observado neste trabalho uma ausência com relação à efetivos experimentos de navegação sobre essa representação de mapa em triângulos de Delaunay.

O trabalho proposto em (CHEN; LIU, 2021) é outro exemplo de mapeamento para navegação a partir da nuvem de pontos esparsa e ruidosa. Para isso, são utilizadas regiões convexas a fim de definir a área livre para navegação. O processo proposto é executado na etapa de pós-processamento. Inicialmente, são adotados pontos dentro da trajetória do agente de forma igualmente espaçados entre si, os quais são usados como ponto de partida para ir expandindo a região convexa pela técnica IRIS (em inglês, *Iterative Regional Inflation by Semidefinite*) (DEITS; TEDRAKE, 2015). Uma visualização dessa técnica, em duas dimensões, é ilustrada na Figura 41. É possível observar que a região da elipse, dado um ponto inicial, vai se expan-

dindo, incrementalmente, até alcançar sua região máxima, definida pelos pontos esparsos que representam os limites de expansão.

Figura 41 – Ilustração da técnica IRIS. Dado um ponto inicial, a elipse começa a se expandir incrementalmente, a qual é limitada pelos pontos esparsos em sua volta.



Fonte: autora.

Antes de aplicar a técnica de forma tridimensional, uma etapa de filtragem na nuvem de pontos é realizada, onde *outliers* são removidos levando em consideração a média da distância entre os pontos 3D vizinhos. Assim, pontos flutuantes e isolados, por exemplo, são removidos. Haverá tantas regiões convexas quantos forem os pontos inciais adotados na curva que representa a trajetória, as quais são fundidas numa próxima etapa adaptada para funcionar com um mapa de regiões convexas.

Para avaliar a técnica, primeiro as nuvens de pontos esparsas são geradas pelo sistema ORB-SLAM2 com dados do sensor estereoscópico, considerando (a) uma base de dados própria em ambiente indoor de pequena escala, e (b) um dataset público de larga escala (KITTI, sequência 05). Em termos de custo computacional, tem-se que o pós-processamento da proposta leva 1.5 segundos para gerar a região livre da cena de pequena escala, e 16.8 segundos na de larga escada. No entanto, os autores pontuam que dividindo esse tempo de processamento para cada ponto adotado na trajetória, o custo ficaria 0.0412 e 0.0723 segundos, respectivamente - caso a técnica funcionasse em conjunto com o sistema SLAM. Mas, vale ressaltar que isso desconsidera a otimização do mapa que é feita durante o mapeamento, o qual induziria correção ou recálculo das regiões geradas.

No trabalho apresentado em (BLOCHLIGER et al., 2018), é utilizada uma câmera monocular e um sensor inercial para gerar, a partir da nuvem de pontos esparsa, um mapa topológico tridimensional associado a um mapa TSDF (explanado na Seção 4.1). O mapeamento do ambiente é feito pelo sistema SLAM visual-inercial apresentado em (LEUTENEGGER et al., 2015). Em seguida, numa etapa de pós-processamento, o mapa e as estimativas de pose são refinados por *bundle adjustment* global (explanada na Seção 3.4.1). Apenas após o mapeamento e a otimização é que a nuvem esparsa resultante é usada como entrada para a técnica proposta apresentada.

Inicialmente, são escolhidos, de forma aleatória, lugares ao longo da trajetória para que, a partir deles, se defina a respectiva região convexa livre. A informação de espaço livre é extraída com o auxílio da biblioteca Voxblox (OLEYNIKOVA et al., 2017). Em seguida, essas regiões são fundidas para alcançar a representação final do mapa TSDF. Para fins de navegação, este

mapa é, então, associado à um mapa topológico[2], cuja representação é a trajetória estimada. Portanto, é difícil obter um mapa topológico que contenha informações de conectividade para áreas que são observadas mas não percorridas fisicamente pelo robô.

Uma das etapas de avaliação consiste no planejamento de rotas dado o mapa gerado e um ponto de destino. Os autores, no entanto, fizeram a adição de mais um sensor, o de laser, para desviar de obstáculos, o que levanta certo questionamento sobre a robustez da proposta apresentada. Uma outra observação é que a técnica consiste em um pós-processamento, onde a entrada é uma nuvem de pontos esparsa já finalizada e otimizada.

Os trabalhos propostos em (STUMBERG et al., 2016) e (ESRAFILIAN; TAGHIRAD, 2016) fizeram uso de uma câmera monocular, o que gera a desvantagem de se desconhecer a escala real do ambiente. Além do mais, testes foram realizados em cenas de pequena escala e voltadas para agentes robóticos capazes de voar. A proposta apresentada em (STUMBERG et al., 2016) mostrou ser uma técnica custosa, que não suporta aplicação em tempo real, e a abordagem trazida em (ESRAFILIAN; TAGHIRAD, 2016) tem sérias limitações quanto a forma dos obstáculos reconhecidos. Os trabalhos descritos em (CHEN; LIU, 2021) e (BLOCHLIGER et al., 2018) são abordagens que consistem em um pós-processamento. De forma geral, todos os trabalhos dessa seção reconstroem o ambiente de maneira tridimensional, que é naturalmente mais custoso que o 2D. Para agentes móveis de chão, a representação 2D do ambiente é suficiente para a navegação. Assim, a abordagem apresentada nessa dissertação aptou pela de modelagem 2D, o OGM, construído com o uso de uma câmera estéreo. Nossa proposta resulta em um baixo consumo de memória, é capaz de realizar o mapeamento globalmente, além de funcionar junto ao sistema SLAM em tempo real.

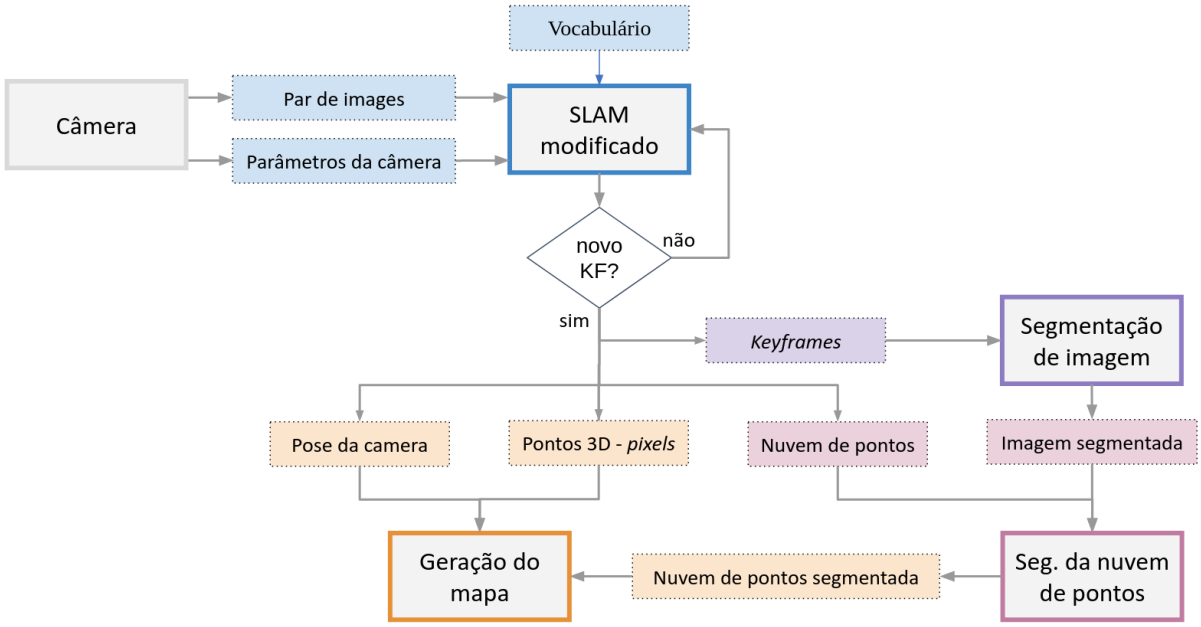
[2]Mapas topológicos são representados por nós distribuídos no ambiente e interligados por restrições/bordas que indicam o caminhos entre esses nós que contém informações métricas adicionais como distância e direção. O planejamento de rotas nessa representação é uma questão de busca gráfica entre os nós.

5 ABORDAGEM PROPOSTA

Neste capítulo, detalhamos toda a estruturação técnica desenvolvida e utilizada na abordagem proposta. Nomeamos nosso sistema de *Segfloor*, o qual tem como finalidade a obtenção de um mapa 2D, o OGM, capaz de indicar a região livre de determinado ambiente através da nuvem de pontos esparsa do sistema SLAM e da segmentação de imagem da área livre.

O sistema é composto de cinco módulos, como mostra a Figura 42. A primeira etapa está relacionada à câmera, isto é, imagens e parâmetros do sistema estereoscópico. Em seguida, o sistema SLAM adotado, o OpenVSLAM, modificado de acordo com a nossa necessidade, gera dados que vão ser usados nos próximos três módulos: o de segmentação da imagem, o da segmentação da nuvem de pontos esparsa, e finalmente, o de geração do mapa de ocupação 2D. Cada um dos cinco módulos é detalhado neste capítulo, mas antes, abordaremos sobre a plataforma responsável por gerir toda a troca de dados do nosso sistema: Robot Operating System (ROS), em inglês, *Robot Operating System*.

Figura 42 – Visão geral da abordagem proposta para obtenção do mapa grade de ocupação.



Fonte: autora.

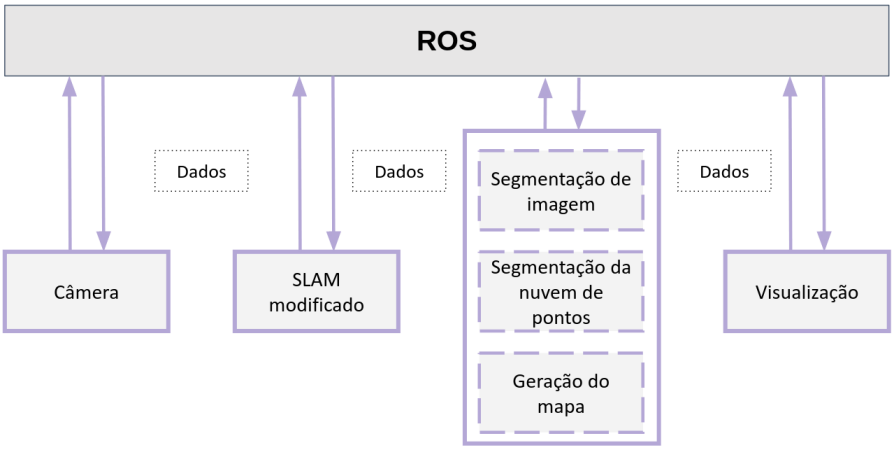
5.1 COMUNICAÇÃO DO SISTEMA VIA ROS

O Sistema Operacional Robótico, ROS, é uma plataforma de código aberto que fornece funcionalidades que facilitam o desenvolvimento robótico e tudo que o cerca. Ele funciona como uma camada de abstração entre software e hardware do robô físico, servindo como um sistema de aplicações e testes, permitindo o desenvolvimento de todo o controle de baixo nível

do robô sem efetivamente a necessidade física do mesmo, além de padronizar e gerenciar toda a troca de dados.

Nossa proposta, assim como a maioria dos projetos voltados para aplicações robóticas, foi desenvolvida sobre a ferramenta ROS. Para isso, os módulos da Figura 42 foram divididos em *processos*, chamados de "nós". A troca de dados se dá via "publicação" e "subscrição" das mensagens, e é gerenciada pelo ROS. O canal de envio da mensagem é definido pelo "tópico", que consiste numa *string* de identificação (canal este que só envia e recebe um único tipo de dado pré-determinado). A Figura 43 mostra a organização dos processos, onde os módulos *segmentação de imagem*, *segmentação da nuvem de pontos* e *geração do mapa* foram alocados em um único processo.

Figura 43 – Visão geral da abordagem proposta para obtenção do mapa grade de ocupação.



Fonte: autora.

5.2 MÓDULO CÂMERA

O módulo "câmera" representa o processo responsável por publicar as imagens direita e esquerda da câmera estéreo, e os parâmetros das câmeras e do sistema, isto é, os parâmetros intrínsecos e extrínsecos (ver Seção 2.3). As imagens enviadas não necessariamente precisam estar retificadas, mas é fundamental que os parâmetros enviados estejam coerentes para a realização da retificação no módulo subsequente (SLAM).

5.3 MÓDULO SLAM

O projeto de código aberto OpenVSLAM consiste em um framework de desenvolvimento de aplicações SLAM. Suas funcionalidades foram chamadas dentro de um processo compatível com ROS. O processo desenvolvido executa o algoritmo de mapeamento e localização no momento em que recebe como entrada (a) as imagens, (b) o parâmetros de configuração da câmera e do sistema, e (c) o arquivo relativo ao vocabulário (responsável pela rápida detecção

de revisita - tópico abordado na Seção 3.3.5). O detalhamento técnico de como exatamente o algoritmo SLAM funciona foi assunto abordado no Capítulo 3, portanto, nesta seção, apenas uma breve recapitulação é descrita.

Inicialmente, de posse das duas imagens, o mapeamento ocorre uma vez que *features* são extraídas e transformadas em pontos 3D através da triangulação (ver Seção 2.2). A localização, no entanto, é estimada a partir da técnica (ver Seção 3.3.3), isto é, pela re-projeção dos pontos 3D na imagem. A nuvem de pontos é otimizada localmente quando um novo *keyframe* é definido através da técnica de *bundle adjustment* (ver Seção 3.4.1), e globalmente quando uma revisita é detectada (ver Seção 3.4.2).

O módulo ROS desenvolvido apresenta as funcionalidades descritas a seguir.

- Sempre que uma imagem é definida como *keyframe* o processo publica o dado de odometria relativo a este *keyframe*;
- Sempre que uma imagem é definida como *keyframe* o processo publica o par de imagens estéreo relativo a este *keyframe*;
- Sempre que uma nova imagem é definida como *keyframe* o processo publica toda a nuvem de pontos esparsa mapeada até então, onde pontos são associados com um identificador único. Para isso, a nuvem informada pelo OpenVSLAM é re-estruturada de acordo com o padrão de mensagem reconhecido pelo ROS. A biblioteca Point Cloud Library (PCL) (RUSU; COUSINS, 2011) (em inglês, *Point Cloud Library*) é utilizada para realizar essa conversão;
- Sempre que uma imagem é definida como *keyframe* o sistema publica, em coordenadas de imagem (par de pixels), as localizações das *features* relativas a este *keyframe*, onde são associados com (a) um identificador único e com (b) sua respectiva representação 3D no espaço (se houver). Em seguida, são re-estruturados de acordo com padrão de mensagem reconhecido pelo ROS. Mais uma vez, a biblioteca PCL é utilizada para auxiliar nessa conversão;
- Os respectivos sistemas de coordenadas do OpenVSLAM e da aplicação de visualização do ROS são diferentes, portanto, tanto a estimativa de odometria quanto os dados relativos aos pontos 3D passam por um processo de conversão do sistema de coordenadas antes do envio dos dados. A matriz de rotação usada é mostrada na Equação 5.1, equivalente a uma rotação de 270^o com relação ao eixo z, e 270^o com relação ao eixo x; $\vec{r}_{(x,y,x)}$ representa a odometria estimada (vetor), que vai da origem do sistema até o ponto (x, y, z), e $\vec{r}_{(x',y',z')}$ indica o novo vetor após a rotação. Não se faz necessária

uma translação sobre o dado de odometria uma vez que a origem do sistema permanece o mesmo, mudando apenas a orientação dos eixos.

$$\vec{r}_{(x',y',z')} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \vec{r}_{(x,y,z)} \tag{5.1}$$

Como se pode perceber pela Figura 42 e pela descrição dos itens acima, os dados só são enviados do módulo SLAM quando um novo *keyframe* é definido. Sua definição passa por uma série de critérios (discutidos na Seção 3.3.4) que evita o consumo de memória com imagens redundantes ou impróprias. Além disso, contribui para o processamento *online* da aplicação. De acordo com testes realizados, a taxa de imagens que requerem processamento no sistema SLAM, quando fazendo a seleção de *keyframes*, caiu de 15 para 2-3 quadros por segundos. Logo, o uso dessa estratégia é estendida para os processos seguintes, permitindo que toda a aplicação continue com o funcionamento *online*. Além do mais, novos pontos 3D só são inseridos na nuvem esparsa quando um *keyframe* é definido (ver Figura 25), portanto, é conveniente que o mapeamento do OGM também só atualize neste momento.

5.4 ESTIMATIVA DO PLANO E SEGMENTAÇÃO DA IMAGEM

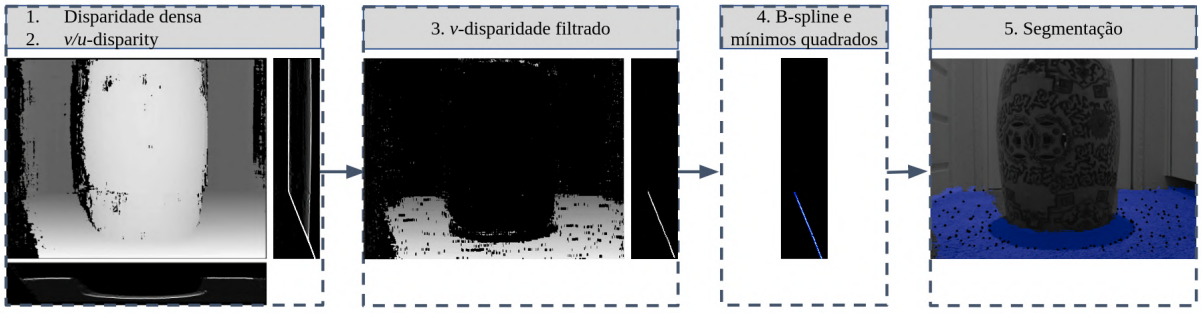
A fim de definir a área livre para navegação, é preciso realizar a estimativa do plano referente ao chão. Diversas propostas se classificam de acordo com o espaço em que a busca pelo plano é executada, isto é, no espaço euclidiano (3D) ou v-disparidade (2D). A primeira, por exemplo, é a usada pelo RTAB-Map, isto é, através dos pontos no espaço tridimensional, é feita uma busca pelo plano que melhor engloba a maioria dos pontos por meio de *plane fitting* (ver Seção 4.1). O segundo caso, é o utilizado neste trabalho, e uma ampla base teórica dessa abordagem pode ser encontrada nos trabalhos descritos em (WEDEL et al., 2009; BROGGI et al., 2013; JASPERS; WUENSCHE, 2014; ZHANG et al., 2018; ZHAO; KATUPITIYA; WARD, 2007).

A Figura 44 ilustra as etapas da estratégia adotada. Dado o par de imagens, o mapa de disparidade é gerado pela técnica SGM (ver Seção 2.6), em seguida, são extraídas as imagens conhecidas como u-disparidade e v-disparidade; obstáculos são filtrados na imagem v-disparidade e o que permanece representa o perfil do plano, o qual é modelado pela técnica *B-spline*, cujos parâmetros são estimados pela técnica dos mínimos quadrados. Por fim, *pixels* são classificados como *chão* ou *não-chão*.

5.4.1 Imagem v/u-disparity

Uma vez que o mapa de disparidade $D(u, v)$ é gerado, é possível extrair duas imagens conhecidas como v-disparidade e u-disparidade as quais representam um histograma relativo aos valores de disparidade. No caso do v-disparidade, são consideradas as linhas de D, enquanto

Figura 44 – Fluxo de processamento da abordagem de segmentação da área livre: mapa de disparidade e o respectivo mapa u/v-disparidade; mapa v-disparidade filtrado; modelagem por *B-spline* e mínimos quadrados, indicada pela curva azul; segmentação da imagem indicada pela região azul.



Fonte: autora.

que, para formar o u-disparidade, as colunas de D são levadas em consideração. Para cada linha (ou coluna) de $D(u, v)$, é calculada a frequência com que os valores de disparidades aparecem, ou seja, é gerado um histograma que considera os *pixels* na horizontal (ou na vertical). A visualização desse histograma é realizada por meio das imagens v/u-disparidade que, quanto mais frequente determinado valor de disparidade em D, maior o grau de intensidade do pixel em $D_{v/u}$.

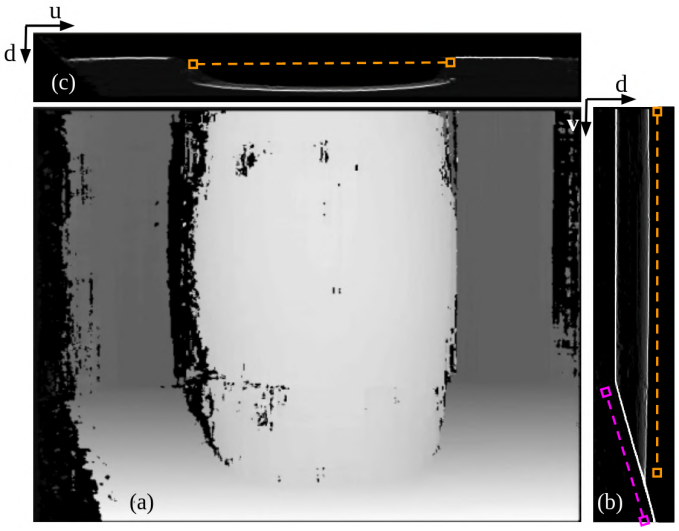
O Algoritmo 1 mostra o processo de cálculo do mapa v-disparidade D_v, onde todo o mapa de disparidade D é percorrido; a intensidade do pixel na imagem D_v é incrementado de acordo com as disparidades em D (valores de disparidade iguais à zero são ignorados pois indicam que o *pixel* correspondente não foi encontrado). No caso do cálculo de u-disparidade, a linha 4 é substituída por $D_u(D(i, j), i) + +$.

Algorithm 1 Cálculo da imagem v-disparidade

Entrada: mapa de disparidade D
Saída: mapa v-disparidade D_v
1: **para** cada i^{th} coluna em D **faça**
2: **para** cada j^{th} linha em D **faça**
3: **se** $D(i, j) > 0$ **então**
4: $Dv(j, D(i, j)) + +$

Na Figura 45.(a) é possível observar um mapa de disparidade e suas respectivas imagens (b) v-disparidade e (c) u-disparidade. Suas formações revelam informações importantes sobre a geometria da cena. Obstáculos são mapeados como um segmento de linha vertical em D_v, e horizontal em D_u, como exemplificado e destacado pelas linhas tracejadas em laranja; enquanto o plano da cena é mapeado como um segmento inclinado, destacado pela linha tracejada rosa.

Figura 45 – Ilustração (a) do mapa de disparidade, (b) da imagem v-disparidade e (c) u-disparidade. Os segmentos tracejados destacam formações que dizem respeito aos elementos da cena.



Fonte: autora.

5.4.2 Remoção de Obstáculos no Mapa v-disparidade

Para efetivamente modelar e ajustar a curva relativa ao plano, é necessário, primeiro, remover demais curvas no mapa v-disparidade que sejam relativas aos obstáculos. Para isso, os valores de disparidade no mapa D são definidos como zero, caso, ao serem comparados com um valor limite a, sejam menores que este limite. O Algoritmo 3 descreve o processo de filtragem de D_v. Um mapa de disparidade relativo ao plano D_{plano} é formado a partir de D, o qual é todo percorrido; se a disparidade do *pixel* em D for maior que a, o *pixel* correspondente em D_{plano} adota o valor zero, caso contrário, recebe o mesmo valor do *pixel* de origem. a é um parâmetro de entrada, o qual adotamos como sendo igual à 5, estimado de forma experimental.

Algorithm 2 Filtrando a imagem v-disparidade

Entrada: mapa de disparidade D; mapa u-disparidade D_u; valor limite a
Saída: mapa de disparidade do plano D_{plano}

1: **para** cada i^{th} coluna em D **faça**
2: **para** cada j^{th} linha em D **faça**
3: **se** $D_u(i, D(i,j)) > a$ **então**
4: $D_{plano}(i,j) = 0$
5: **senão**
6: $D_{plano}(i,j) = D(i,j)$

A Figura 46 traz o resultado visual desse processo. Todos os *pixels* no mapa de disparidade D são comparados com um valor limite a, e os que estão acima desse limite são ignorados. Dessa forma, os obstáculos são removidos de D, como mostra a Figura 46.(c), e um novo processo de cálculo de v-disparidade (Algoritmo 1) é realizado considerando esse

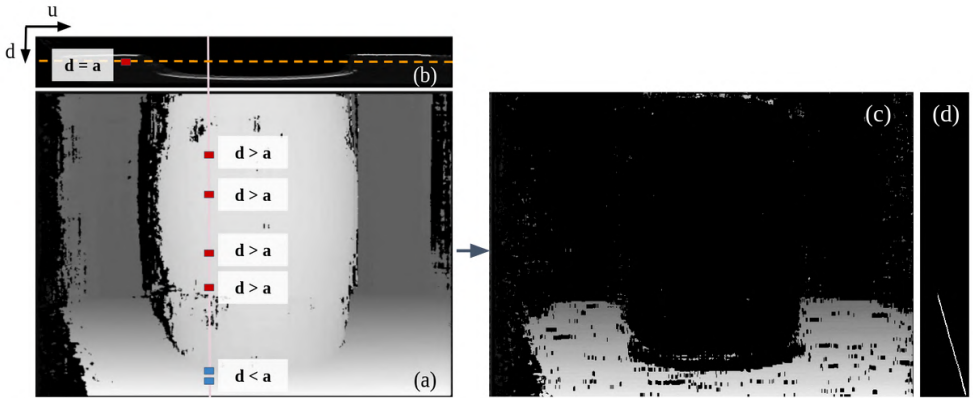
Algorithm 3 Filtering the v-disparity map

Entrada: disparity map D; u-disparity map D_u; threshold a
Saída: disparity map related to the plane D_{plane}

```
para each i^th column in D faça
    para each j^th line in D faça
        se D_u(i, D(i,j)) > a então
            D_plane(i,j) = 0
        senão
            D_plane(i,j) = D(i,j)
```

mapa resultante D_{plano}. Na Figura 46.(d) é possível ver a imagem v-disparidade sem a linha vertical proveniente do obstáculo (ver Figura 45.b). Uma vez que se tem o resultado de D_v filtrado, a modelagem do plano pode ser efetivada.

Figura 46 – Ilustração do processo de filtragem da imagem v-disparidade. Em (a) se tem o mapa de disparidade; (b) a imagem u-disparidade; (c) o mapa de disparidade após remoção de determinados *pixels*; (d) a nova imagem v-disparidade filtrada referente à (c).



Fonte: autora.

5.4.3 Modelagem do Plano

Os *pixels* de alta intensidade na imagem v-disparidade D_v filtrada formam, visualmente, uma curva que representa o plano. Sua modelagem, neste trabalho, é realizada pela curva B-spline, que permite definir formas ou superfícies complexas, sejam elas planas, inclinadas e até onduladas, todas presentes em cenários reais.

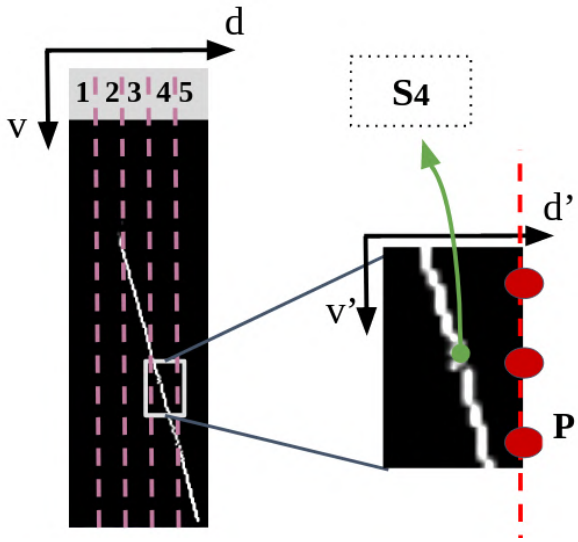
Uma função B-spline consiste na representação de uma curva formada pela combinação de *splines*, que por sua vez são equações polinomiais de ordem n. Neste trabalho, utilizamos *splines* de ordem cúbica, ou seja, possuem a forma genérica demostrada na Equação 5.2. Como a ordem do polinômio indica a quantidade de vezes que a respectiva curva consegue mudar de sentido, *splines* de ordem cúbica cobrem as possíveis situações do contexto real, por exemplo,

o caso de rampas.

$$S(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \tag{5.2}$$

Foram utilizadas cinco *splines*, dividindo a imagem v-disparidade D_v em cinco seções igualmente espaçadas (B-spline uniforme), como ilustrado na Figura 47. Dessa forma, em D_v, a abscissa d dos pontos de controle P são conhecidas e fixas, e apenas as ordenadas v mudam de acordo com os cálculos de aproximação da curva S estimada (os pontos destacados em vermelho na Figura 47 são utilizados para ilustrar três possíveis posições para P). Ainda na Figura 47, é destacada uma dessas seções relativa à *spline* S_4. Uma vez que a curva esteja modelada nesse trecho, é interessante observar que a equação só é válida para o mesmo intervalo de d. Em um processo iterativo, o Estimador-M (HUBER, 1992; QINGGUO; LONGSHENG, 2008; ZOU; ZHU, 2014) estima os parâmetros das *splines* e pondera as medições de acordo com o quão bem elas são capazes de representar o modelo feito.

Figura 47 – A imagem v-disparidade é divida em 5 *splines* S, destacadas pelas linhas tracejadas em rosa. Uma dessas seções (S_4) é evidenciada, onde os pontos em vermelho exemplificam três possíveis posições para o respectivo ponto de controle P.



Fonte: autora.

5.4.4 Segmentação da Imagem

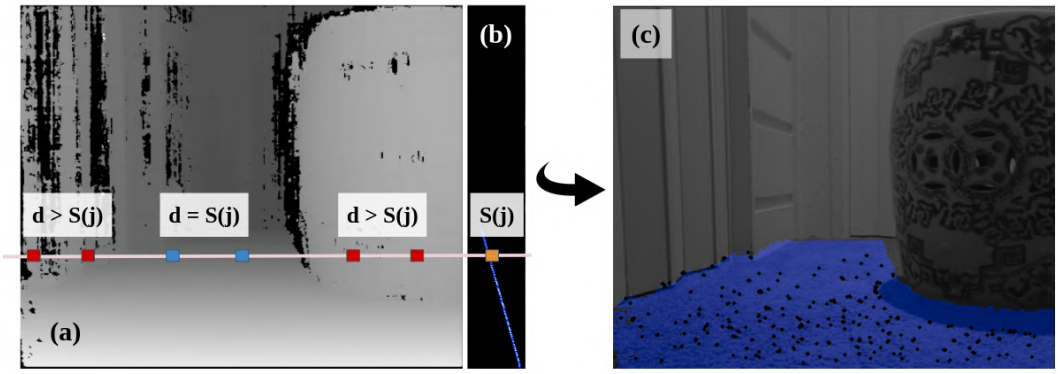
A curva azul na Figura 48.(b) indica a função B-spline cujos parâmetros foram estimados a partir do mapa v-disparidade filtrado a partir da técnica de mínimos quadrados. Ela é utilizada como função de consulta para cada valor d de disparidade do mapa D. O algoritmo 4 descreve as etapas desse cálculo: toda a imagem D é percorrida; se o respectivo valor de um pixel qualquer $D(i, j)$ estiver sobre o valor determinado pela curva B-spline (considerando uma tolerância indicada por th), então, ele é classificado como *chão*, caso contrário, como *não-chão*.

Algorithm 4 Cálculo da imagem segmentada

Entrada: mapa de disparidade D
Saída: imagem segmentada S
1: **para** cada i^{th} coluna em D **faça**
2: **para** cada j^{th} linha em D **faça**
3: **se** $D(i,j) = B(j) \pm th$ **então**
4: $S(i,j) \leftarrow chão$
5: **senão**
6: $S(i,j) \leftarrow não - chão$

A Figura 48 ilustra esse processo. Após consultar todos os *pixels* do mapa de disparidade e classificá-los como chão ou não-chão, a classificação é estendida para a imagem esquerda do par de imagens estéreo, ilustrada na Figura 48.(c).

Figura 48 – Ilustração da segmentação da imagem. Valores de disparidade no (a) mapa de disparidade são consultados pela (b) curva modelada S para definir a classificação do pixel entre chão e não-chão, onde (c) mostra o resultado sobre a imagem esquerda do par estéreo.



Fonte: autora.

5.5 SEGMENTAÇÃO DA NUVEM DE PONTOS

O módulo relativo à segmentação da nuvem de pontos requer os seguintes dados (ver Figura 42): a nuvem de pontos esparsa global; a imagem segmentada; *features* em coordenadas de imagem e de mundo com relação ao *keyframe* atual. O módulo ROS desenvolvido apresenta as funcionalidades descritas a seguir.

- Sempre que uma imagem é definida como *keyframe* no sistema SLAM, o processo subscreve toda a nuvem de pontos esparsa mapeada até então, onde pontos foram associados a um identificador k único. A nuvem PC é re-estruturada em uma estrutura de dados do tipo chave-valor, que permite rápido acesso e manipulação dos dados, onde os identificadores k foram usados para definir as chaves, e os pontos 3D para definir os valores;

- Sempre que uma imagem é definida como *keyframe* no sistema SLAM, o processo subscreve, em coordenadas de imagem (par de pixels), as localizações das *features* $F_k(i, j)$ (relativas a este *keyframe*), e sua respectiva representação 3D no espaço $F_k(x, y, z)$ (se houver). k representa o identificador único;
- Sempre que uma imagem é definida como *keyframe* no sistema SLAM, o processo subscreve o par de imagens relativo a este keyframe, gerando a imagem segmentada do mesmo (Seção 5.4).

Todo o conjunto de *features* $F_k(i, j)$ dessa iteração é verificado, se sua posição (i, j) estiver numa região classificada como chão, $F_k(x, y, x)$ também é definido como chão. A estrutura de dados dos pontos 3D PC_{seg} que indicam chão é, então, incrementada. Caso contrário, $F_k(x, y, x)$ é inserido na estrutura de dados dos pontos 3D PC_{obs} que indicam obstáculos. Esse processo é ilustrado no Algoritmo 5. É importante citar que a atualização da nuvem de pontos do sistema SLAM é igualmente replicada neste módulo graças ao identificador único k.

Algorithm 5 Cálculo da imagem segmentada

Entradas: $F_k(i, j) \leftarrow$ features do keyframe atual em coordenadas de pixel
$F_k(x, y, z) \leftarrow$ features do keyframe atual em coordenadas de mundo
$A_{seg} \leftarrow$ região segmentada
Saídas: $PC_{chao} \leftarrow$ nuvem de pontos classificados como chão
$PC_{obs} \leftarrow$ nuvem de pontos classificados como obstáculo

```
1: para cada k faça
2:     se F_k(i, j) ∈ A_seg então
3:         PC_chao ← F_k(x, y, z)
4:     senão
5:         PC_obs ← F_k(x, y, z)
```

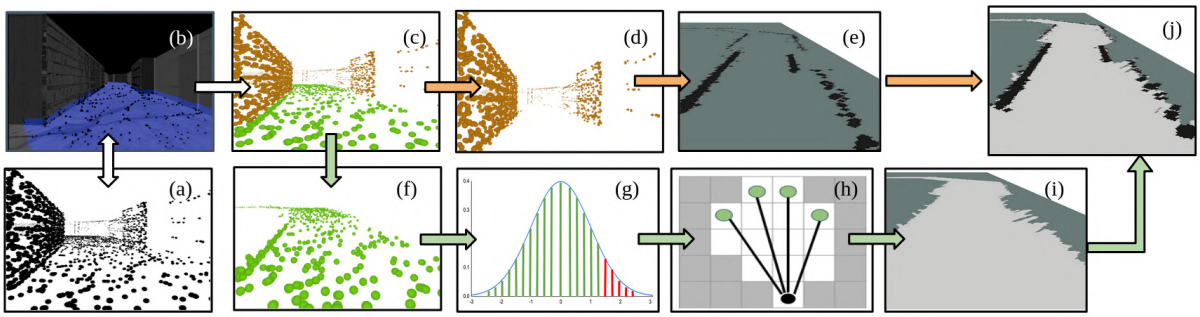
5.6 GERAÇÃO DO MAPA 2D

Como descrito na Seção 1.1, o mapa do tipo grade de ocupação binário é formado por regiões uniformemente espaçadas, onde cada uma dessas regiões é indicada como ocupada, livre de obstáculos ou indefinida. O tamanho do espaçamento é definido pela resolução escolhida, isto é, quanto maior o seu valor, mais informação do ambiente se têm.

O módulo relativo à geração do OGM 2D requer os seguintes dados (ver Figura 42): a nuvem de pontos segmentada; as *features* em coordenadas de imagem e de mundo com relação ao *keyframe* atual; e a estimativa de pose da câmera também desse *keyframe*. O processo completo é ilustrado na Figura 49.

Inicialmente, com a associação da nuvem de pontos geradas pelo sistema SLAM (Seção 5.3) e a segmentação do plano (Seção 5.4), a nuvem de pontos é segmentada entre PC_{obs} e $PC_{chão}$ (Seção 5.5). O mapa de ocupação final é o resultado da soma de dois OGMs. O

Figura 49 – Fluxo de processamento do módulo *geração do mapa*. Com a nuvem de pontos (a) e a imagem segmentada (b), a nuvem de pontos também é segmentada (c). Os pontos classificados como não-chão (d) preenchem a primeira camada (e); Em relação aos pontos definidos como chão (f), um filtro gaussiano (g) é aplicado considerando as distâncias do raio; As distâncias válidas são projetadas em uma segunda camada (h). O OGM final em (i) é composto pela soma dessas duas camadas.

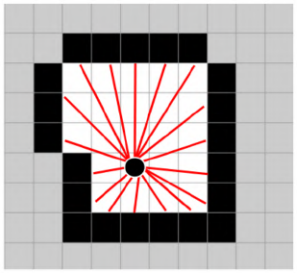


Fonte: autora.

primeiro consiste na projeção ortogonal de todos os pontos 3D existentes em PC_{obs}, ilustrado na Figura 49.(e). Todos os pontos projetados definem grades "ocupadas".

Para formar o segundo mapa, a princípio é realizada a projeção dos pontos em $PC_{chão}$ no plano 2D, mas apenas àqueles que são visualizados pelo *keyframe* da iteração atual. Eles definem grades livres, porém, como é um dado esparso, apenas sua projeção não é suficiente para cobrir a região navegável. Por isso, a mesma estratégia usada em sensores de alcance, chamada de *ray tracing*, é adotada. No caso do *laser*, por exemplo, através do *ray tracing* a região livre é determinada pela distância do raio que vai da posição do sensor até o obstáculo atingido pelo feixe, como ilustrado na Figura 50. Esta inclusive, é a mesma estratégia utilizada pelo RTAB-Map, onde os raios são definidos da posição atual da câmera até todos os voxels/grades ocupados visíveis no momento (abordado na Seção 4.1). No nosso caso, o raio é formado entre a posição atual do agente e todos as grades *livres* visíveis pelo *keyframe* em questão, como mostra a Figura 49.(h).

Figura 50 – Ilustração da técnica *ray tracing* para mapeamento de região livre, neste caso fazendo uso de um sensor de alcance (*laser*).



Fonte: autora.

No entanto, antes de aplicar o *ray tracing*, adotou-se um filtro para remover potenciais *outliers*. Sabe-se que, em estatística, um quartil Q é um dos três valores que divide o conjunto de dados em quatro partes iguais, onde Q_1 e Q_3 representam, respectivamente, o quartil inferior e superior. A amplitude interquartil IQR é cálculada por $Q_3 - Q_1$. De acordo com a literatura, *outliers* podem ser definidos através do limite superior th_{sup} e inferior th_{inf}, formalizado pela Equação 5.3. Ou sejam valores maiores que th_{sup} e menores que th_{inf} são interpretados como *outliers* (MILLER, 1993; ROUSSEEUW; HUBERT, 2011).

$$\begin{aligned} th_{inf} &= Q_1 - 1.5IQR \\ th_{sup} &= Q_3 + 1.5IQR \end{aligned} \tag{5.3}$$

Para calcular os valores dos quartis Q_1 e Q_3, assumimos que as distâncias dos raios (formados entre a pose da câmera e cada grade livre) seguem o comportamento de uma curva normal, formalizada pela Equação 5.4 e ilustrada na Figura 49.(h). A letra μ representa a média aritmética simples das distâncias, e σ representa o desvio padrão.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}^{-\left(\frac{x-\mu}{2\sigma}\right)^2} \tag{5.4}$$

Dessa forma, Q_1 e Q_3 podem ser calculados como formalizado na Equação 5.5. Por fim, toda distância d maior do que th_{sup} é considerado um raio *outlier* (apenas o limite superior foi usado).

$$\begin{aligned} Q1 &= \mu - 0.675\sigma \\ Q3 &= \mu + 0.675\sigma \end{aligned} \tag{5.5}$$

Os raios que não são removidos na etapa de filtragem são efetivamente gerados no OGM, projetados como região livre, como ilustra a Figura 49.(i). Finalmente, ambos os mapas (o OGM de obstáculos e o OGM de regiões livres), os quais possuem a mesma orientação, posição, tamanho, escala e resolução, são somados, resultando na versão final do mapa de ocupação 2D, como mostra a Figura 49.(j).

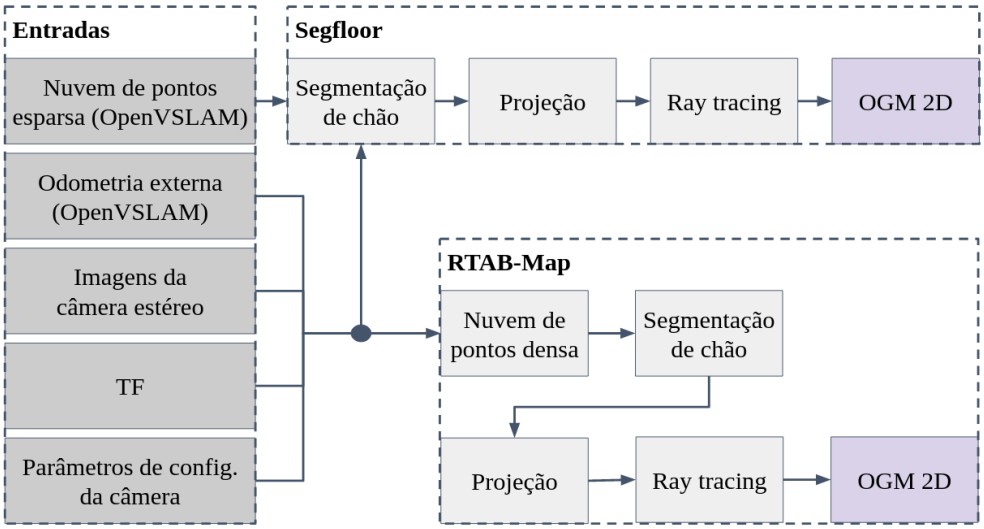
6 AVALIAÇÃO

A avaliação do OGM gerado pode ser desafiadora uma vez que as abordagens variam quanto a estratégia de validação da técnica. Considerando os trabalhos relacionados apresentados na Seção 4 (apenas aqueles que geram um OGM 2D ou 3D), foi observado que nas propostas descritas em (YAN; LAN; YANG, 2020; YANG et al., 2017; SCHAUWECKER; ZELL, 2014) e no projeto de código aberto RTAB-Map (LABBÉ; MICHAUD, 2019), a avaliação do mapa se dá por uma discussão qualitativa sobre os resultados visuais do mapa, além da análise de métricas computacionais, como tempo de processamento e uso de memória. Já nos trabalhos descritos em (XU et al., 2019; STUMBERG et al., 2016; ESRAFILIAN; TAGHIRAD, 2016) o custo computacional não foi reportado. Vale ressaltar que em todos eles, outras medidas, que não diretamente relacionadas ao OGM em si, também foram avaliadas. Por outro lado, em ambientes de competição robótica, o mapa *ground truth* é fornecido para que seja sobreposto ao OGM 2D gerado pelas equipes (PELLENZ; PAULUS, 2008; BALAGUER et al., 2009). Nesses trabalhos, uma avaliação qualitativa mais confiável foi formada pela comparação dos contornos do mapa e das semelhanças locais e globais.

O trabalho de Collins (COLLINS; COLLINS, 2007) apresenta uma proposta de *benchmark*, a qual é usada em seu trabalho para avaliar OGMs 2D que foram gerados por cinco técnicas diferentes com o uso de sonar. Em termos de análise quantitativa, uma métrica de precisão é calculada por meio de possíveis trajetos, classificados entre verdadeiro-positivos ou falso-positivo dado o OGM *ground truth*, previamente definidos a partir de um mapa de rotas (em inglês, *roadmap*) desenvolvido pelo diagrama de Voronoi no mapa gerado. O trabalho descrito em (MELLO et al., 2020) é outro exemplo que avalia os OGMs 2D gerados por duas propostas a partir de um mapa de rotas e realizando atividades de navegação.

As métricas adotadas neste trabalho, especificadas na Seção 6.1, foram comparadas com o trabalho RTAB-Map. De acordo com nossas pesquisas, este é o projeto mais semelhante à nossa proposta, que por ser de código aberto, nos permite gerar resultados comparativos. Conforme detalhado na Seção 4.1 - Figura 38, este projeto pode ser executado com informações de odometria externa para fornecer o OGM, ou gerar o dado pelo próprio sistema SLAM. Nós usamos a configuração mostrada na Figura 51, isto é, adicionamos dados de odometria provenientes do OpenvSLAM, além de todos os outros dados requeridos, isto é, as imagens direita e esquerda, os parâmetros de configuração da câmera e as transformações (TF) dos sistemas de coordenadas envolvidos. Para a nossa proposta, a entrada relativa à nuvem de pontos esparsa (gerada pelo sistea SLAM) também é usada. É possível perceber que a estratégia de criação do OGM 2D do RTAB-Map passa por uma reconstrução densa do ambiente, enquanto a técnica proposta nessa dissertação faz isso diretamente da nuvem esparsa.

Figura 51 – Ilustração dos dados de entrada requeridos pelas técnicas Segfloor e RTAB-Map, bem como o fluxo de processamento de cada uma dessas abordagens para gerar o OGM 2D. O uso da nuvem densa e a estratégia de definição da região livre são as principais diferenças.



Fonte: autora.

6.1 MÉTRICAS DE AVALIAÇÃO

Considerando os trabalhos mencionados, avaliamos quais e como as possíveis métricas se adequariam à abordagem proposta, detalhando-as a seguir.

6.1.1 Sobreposição, Precisão e Cobertura do Mapa

A primeira avaliação adotada é uma métrica qualitativa dada pela sobreposição dos mapas. Dois tipos de bancos de dados foram gerados para obter os mapas *ground truth*, um considerando cenas de um ambiente real e outro considerando uma cena gerada sinteticamente, discutidas nas Seções 6.2.1 e 6.2.2.

Para efetivamente realizar uma avaliação quantitativa, adotamos uma técnica de mapa de rotas probabilístico (Probabilistic Road Map (PRM)), um grafo de rede no qual os nós são distribuídos aleatoriamente sob um determinado OGM, e as conexões entre eles representam as possibilidades do caminho. Esta abordagem, também usada nos trabalhos (COLLINS; COLLINS, 2007; MELLO et al., 2020), reflete a precisão do mapa considerando seu propósito: navegação. A distribuição de nós segue uma distância mínima dos outros, e sua quantidade foi aumentada até que toda a área de navegação fosse povoada por opções de caminhos.

Duas interpretações podem ser consideradas. A primeira, a precisão, é calculada gerando o PRM no mapa estimado, e replicando-o no *ground truth*. Os caminhos que induziriam a uma colisão representam caminhos falsos-positivo (FP). Elementos estimados como caminhos corretamente representam os verdadeiros-positivo (VP). A Equação 6.1 mostra como o cálculo é realizado. Portanto, a precisão diz respeito aos dados que realmente são positivos conside-

rando todos os dados *estimados* como positivo. Essa métrica é fundamental para garantir a integridade física do robô.

$$P = \frac{VP}{VP + FP} \tag{6.1}$$

Por outro lado, aplicando inicialmente o PRM no mapa ground truth e replicando-o no mapa estimado, é possível calcular a métrica *recall*. *Recall* indica qual a porcentagem de dados classificados como positivos comparado com a quantidade real de *todos* os caminhos positivos no *ground truth*. Os elementos falsos-negativo (FN) são justamente os caminhos verdadeiros no *ground truth* e falsos no mapa gerado. A Equação 6.2 mostra como o cálculo é realizado. Neste caso, essa métrica informa com relação à cobertura do mapa.

$$R = \frac{VP}{VP + FN} \tag{6.2}$$

Um mapa gerado que apresenta boa precisão, mas baixo *recall*, por exemplo, garante a integridade física do robô, porém não consegue navegar por toda a região livre por falta de caminhos no grafo (baixa cobertura).

6.1.2 Desempenho Computacional e Memória

Nosso objetivo foi em propor uma abordagem que mapeie com precisão, porém de forma rápida e com baixo consumo de memória. Avaliamos o tempo de processamento de forma a ilustrar como o uso de uma nuvem de pontos esparsa representa uma alternativa rápida frente às abordagens baseadas em mapeamento denso. Além disso, um extenso percurso foi usado para extrair o desempenho computacional do Segfloor, levantando métricas relacionadas ao tempo de processamento e uso de memória, ambas em função do número de atualizações do mapa. Dessa forma, foi possível avaliar como a técnica se comporta frente à cenas de larga escala. A mesma avaliação foi feita considerando o projeto RTAB-Map para fins de comparação.

6.2 BASE DE DADOS

A fim de validar nossa proposta em um cenário real, tentamos inicialmente usar uma base de dados pública, especificamente a desenvolvida por pesquisadores do MIT (FALLON et al., 2013), que foi a mesma utilizada pelo trabalho RTAB-Map na geração dos mapas de ocupação. No entanto, como ilustrado na Figura 52, as imagens extraídas apresentavam baixa qualidade e resolução, comprometendo consideravelmente o mapa de disparidade (Figura 52.c), e por consequência, a avaliação da técnica proposta. Outro fator observado nessa base de dados é que o chão não tem textura, um pré-requisito do nosso sistema, mais especificamente da técnica de disparidade densa SGM (apresentada na Seção 2.6).

Figura 52 – Base de dados do MIT, cujas imagens são de baixa qualidade, gerando um mapa de disparidade ruidoso.

Fonte: autora.

Diante da problemática, optamos pelo uso de outra base de dados pública. Adicionalmente, também geramos imagens próprias, tanto em ambiente real quanto em ambiente sintético, detalhados nas Seções 6.2.1 e 6.2.2, respectivamente.

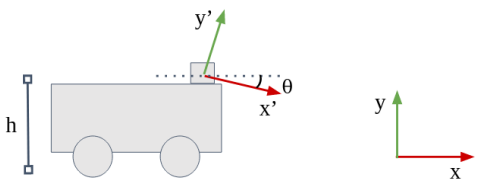
6.2.1 Base de Dados em Ambiente Real

Uma segunda opção de cenário real, com proposta de ambiente bastante similar ao do desenvolvido pelo trabalho descrito em (FALLON et al., 2013), e igualmente extenso, foi considerada: a base de dados *Rawseeds* (CERIANI et al., 2009). Apenas um pequeno trecho continha chão com textura, se adequando ao pré-requisito da proposta. Resultados são apresentados considerando trechos com e sem textura, onde os mapas gerados pelo Segfloor e pelo RTAB-Map são apresentados no Capítulo 7.

A fim de ampliar nossos testes em ambiente real, optamos por desenvolver nossa própria base de dados. A câmera estéreo utilizada é do modelo ELP-1080P2CAM-L28[1]. Imagens foram gravadas em resolução 640x480, com frequência de registro de 15 Hz. O posicionamento da câmera com relação ao sistema de coordenadas fixo é descrito pela Equação 6.3; a Figura 53 é usada de forma complementar, onde $\theta = 7°$ e $h = 21cm$. Os parâmetros intrínsecos e extrínsecos da câmera foram gerados pela aplicação ROS.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.3}$$

Figura 53 – Relação do sistema de coordenadas fixo com o sistema de coordenadas da câmera.



Fonte: autora.

[1]Câmera utilizada disponível em: https://bit.ly/3qVPThX

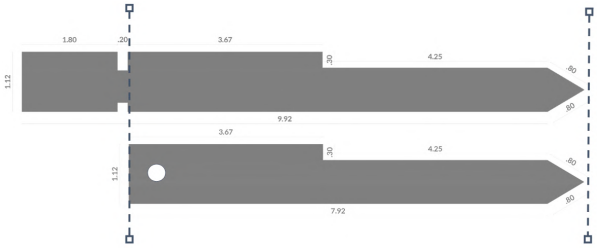
Quadros de imagem relativos aos ambientes filmados são ilustrados na Figura 54. É possível observar um chão texturizado e, principalmente para o *corredor 1* e *2*, paredes com pouca textura. Na cena *complexa* adicionamos elementos de tamanhos, formas e texturas diversos. No *corredor 1*, uma porta semi-aberta forma uma pequena passagem para a continuidade da cena. Já no *corredor 2*, essa porta aparece fechada, e com um obstáculo cilíndrico ao final.

Figura 54 – Cenas em ambiente real utilizadas para avaliação da técnica proposta.

Fonte: autora.

Mapas *ground truth* foram gerados nas cenas cujas delimitações estivessem bem definidos, de forma a garantir medições (manuais), com precisão. Portanto, apenas as cenas *corredor 1* e *2* apresentam mapas *ground truth*. A Figura 55 mostra a planta baixa com as respectivas medidas em metros, onde o segmento tracejado indica a região coincidente. Para essas cenas (além da sintética, abordada na Seção 6.2.2), foram realizadas as avaliações propostas na Seção 6.1.1, isto é, sobreposição e precisão do mapa em termos de navegação. Além disso, todas as cenas foram visualmente observadas e discutidas como parte da avaliação qualitativa.

Figura 55 – Mapas *ground truth* relativos aos corredores 1 e 2, com as respectivas medidas em metros.



Fonte: autora.

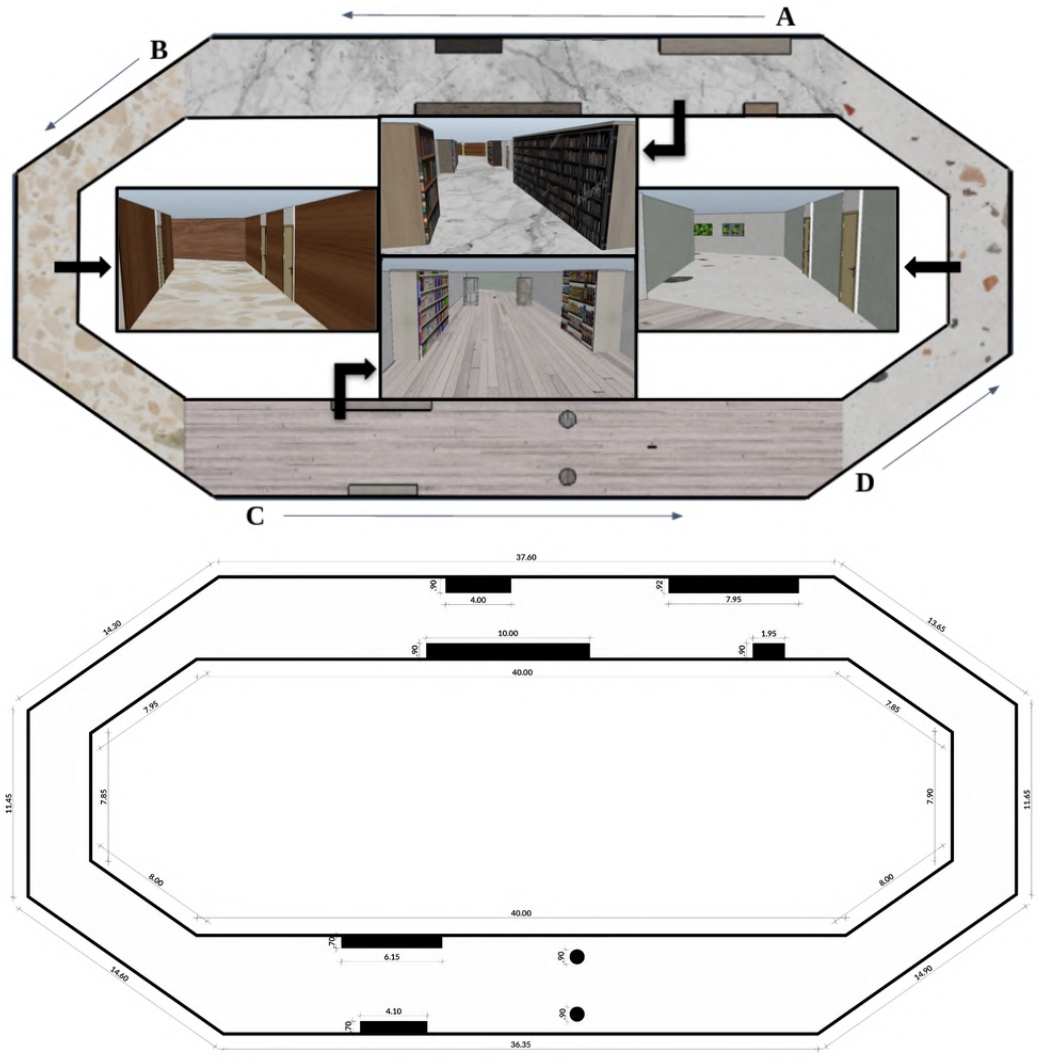
6.2.2 Base de Dados em Ambiente Sintético

A fim de avaliar a técnica proposta frente a uma cena de larga e escala, e dado as limitações discutidas na Seção 6.2 com relação às possíveis bases de dados públicas, optamos por desenvolver uma cena em ambiente virtual. Adotamos o simulador V-REP por apresentar

considerável grau de liberdade para edição de cenas comparado com outras alternativas (MELO et al., 2019).

A Figura 56 mostra a vista superior da cena criada e o respectivo mapa ground truth com as medidas em metros. Para cada trecho há uma imagem frontal associada. É possível perceber que buscamos variar espessuras do corredor e texturas tanto do chão quanto das paredes. No segmento CDA, por exemplo, paredes apresentam textura mais uniforme. No segmento AB, prateleiras de livros foram adicionadas e, em CD, prateleiras de supermercado e pilastras foram acrescentadas. Objetos foram adicionados de forma a reduzir o máximo de pontos cegos para a câmera, o que resultaria em imprecisão no mapeamento. A cena criada apresenta uma área total de $628.7m^2$ e uma trajetória percorrida de 138.4 metros.

Figura 56 – Vista superior da cena sintética criada. Pada cada trecho, é ilustrado uma vista frontal. Na planta baixa, medidas do mapa são apresentadas em unidades de metros.



Fonte: autora.

O posicionamento da câmera com relação ao sistema de coordenadas fixo é ilustrado

pela Figura 53, onde $\theta = 0°$ e $h = 40cm$. A frequência de registro das imagens foi de $15Hz$, com a resolução de 640x480. Os parâmetros intrínsecos e extrínsecos da câmera foram facilmente definidos por se tratar de uma câmera "perfeita". A definição não trivial é quanto ao comprimento focal, para isso, utilizamos as Equações 6.4 e 6.5, onde FoV (em inglês, *field of view*) é o ângulo de abertura (usamos 90^o); e r é a relação X/Y, em que X e Y é o número de colunas e linhas da imagem, respectivamente (resolução). Uma vez calculados os ângulos de abertura nas direções x e y, isto é, FoV_x e FoV_y, foi possível encontrar o respectivo comprimento focal f_x e f_y.

$$\begin{aligned} FoV_x &= FoV \\ FoV_y &= 2tan^{-1}\left[tan\left(\frac{FoV}{2}\right)/r\right] \end{aligned} \tag{6.4}$$

$$\begin{aligned} f_x &= \frac{X}{2} \\ f_y &= \frac{Y}{2}tan\left(\frac{FoV}{2}\right)cot\left(\frac{FoV_y}{2}\right) \end{aligned} \tag{6.5}$$

7 RESULTADOS

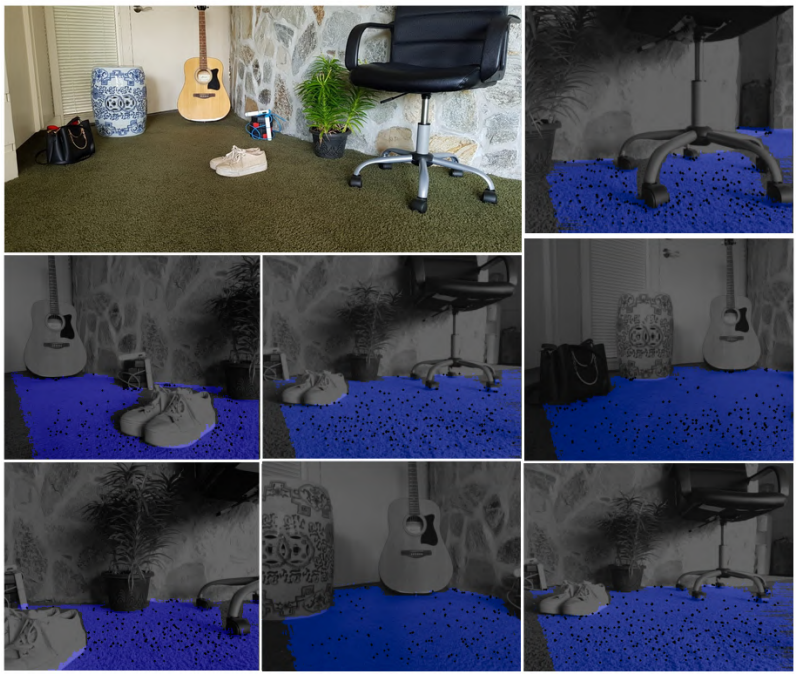
Com base na metodologia de avaliação, abordada no Capítulo 6, os resultados serão apresentados e discutidos no presente capítulo. É avaliado cada ambiente por vez, com as respectivas métricas cabíveis. Por fim, é apresentada a sobreposição de mapas das cenas que possuem *ground truth*, junto com o resultado da precisão dos OGMs considerando o mapa de rotas probabilístico - PRM.

As avaliações que dizem respeito às métricas computacionais foram executadas em um computador com sistema operacional Ubuntu 18.04, processador Intel Core 2.60 GHz i7-9750H e 16 GB de RAM. Para fins ilustrativos, é disponibilizado um vídeo (MELO, 2021) da aplicação.

7.1 CENA COMPLEXA

A cena que denominamos de *complexa*, consiste em vários objetos espalhados, de tamanhos diversos. A Figura 57 traz uma visão geral da cena junto à sete quadros de imagem do ambiente com a respectiva segmentação do chão destacado em azul.

Figura 57 – Visão geral da cena *complexa* e de demais imagens com a região segmentada em azul. Pontos em azul escuro indicam as *features* detectadas pelo sistemas SLAM e que foram segmentadas como "chão" por nossa abordagem.



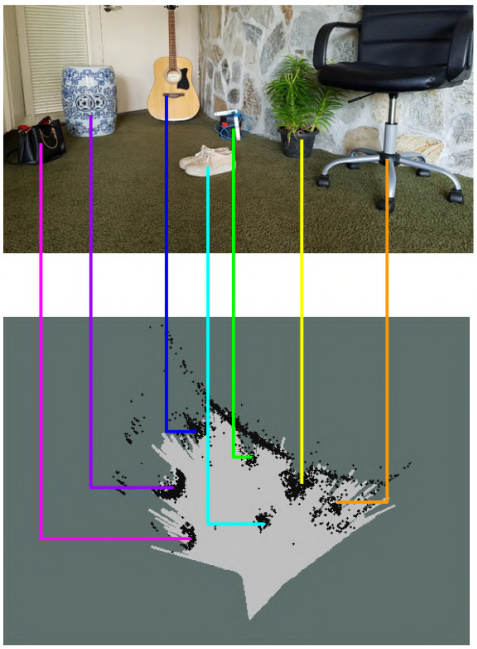
Fonte: autora.

Na Figura 57, é possível perceber o bom resultado na segmentação até mesmo para os suportes da cadeira ou ainda, na presença de até quatro objetos, apresentando o contorno dese-

jado. Na região destacada, há também pontos em azul escuro, que são as *features* detectadas pelo sistemas SLAM e que foram definidas como "chão" por nossa abordagem, posteriormente utilizadas para o *ray tracing*[1]. A classificação dessas *features* também mostram um bom resultado.

A Figura 58 mostra o resultado do OGM 2D gerado por nossa proposta, onde a resolução da grade adotada foi de $0.01m^2$. Nela, há também a relação entre objetos da cena real com as respectivas regiões definidas como "ocupadas". É possível ver que todos os obstáculos foram devidamente reproduzidos. Adicionalmente, graças à correta segmentação da imagem (ilustrada na Figura 57), a região navegável foi ricamente indicada e está visualmente coerente.

Figura 58 – Relação dos objetos na cena real com as regiões definidas como "ocupadas" no OGM 2D gerado por nossa proposta.



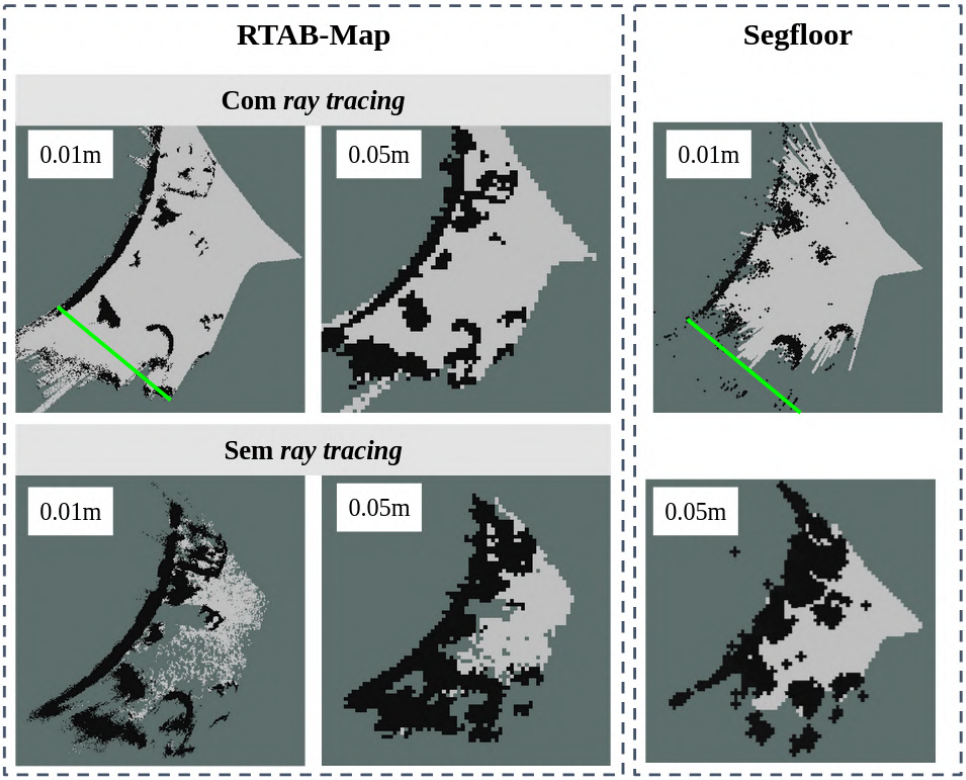
Fonte: autora.

Comparamos nosso mapa com o gerado pelo RTAB-Map nesta mesma cena. Tem-se que a resolução da grade 2D padrão e recomendada pelo autor é de $0.05m^2$, no entanto, por se tratar de uma cena de pequena escala e com objetos menores, optamos por testar com uma resolução menor, de $0.01m^2$, para que mais detalhes pudessem ser percebidos e replicados no mapa. A Figura 59 mostra os OGMs resultantes usando o Segfloor para essas duas resoluções, e usando o RTAB-Map para quatro configurações: variando o parâmetro de resolução, e a opção de mapeamento com ou sem *ray tracing*.

Observando os mapas que variam em resolução, é possível perceber que os mapas obtidos com resolução menor apresentam curvas e limitações mais bem definidas, com maior fidelidade aos detalhes. Quando as grades adotam o valor de $0.05m^2$, a região que indica obstáculos tende

[1] Na nossa abordagem, *ray tracing* é o feixe de raio que vai da posição atual da câmera até todos as grades do OGM livres e visíveis pelo *keyframe* atual.

Figura 59 – OGMs gerados pelo RTAB-Map, considerando variação da resolução e da opção de mapeamento com ou sem *ray tracing*; e OGMs gerados pelo Segfloor para resoluções diferentes.



Fonte: autora.

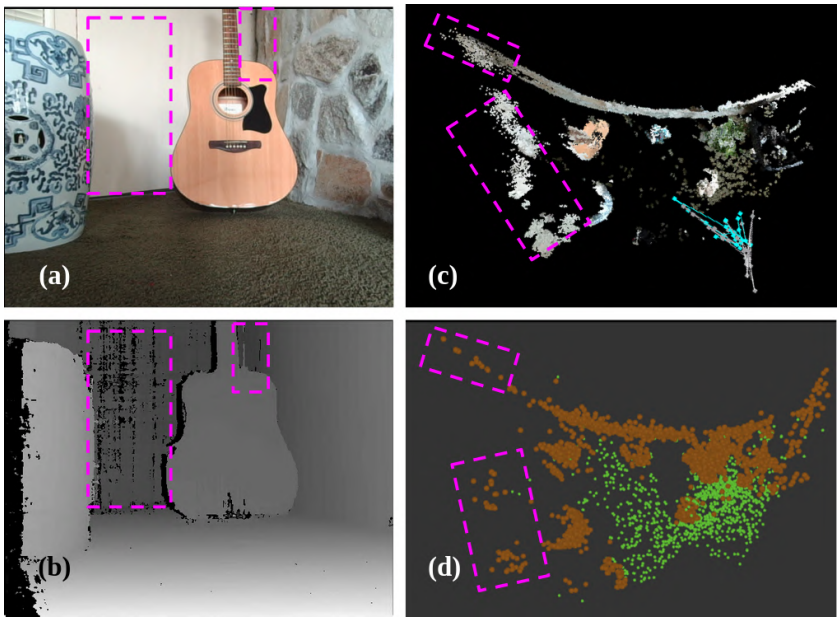
a aumentar seus contornos, ocupando mais espaço do que realmente ocupa. Avaliando pelo aspecto de segurança do agente robótico, este caso pode ser mais recomendado, de forma a evitar possíveis colisões. Com relação aos mapas sem *ray tracing* gerados pelo RTAB-Map, tem-se que, embora o chão seja texturizado, os OGMs apresentam a região navegável de forma esparsa, especialmente quando as grades são de $0.01m^2$. De forma geral, todos os mapas conseguem perceber e indicar as regiões ocupadas como desejável.

Ainda na Figura 59, foi destacada a estimativa de uma das delimitações do ambiente por um segmento em verde. Essa definição é baseada pela foto da cena, onde o violão está encostado na porta e o banco cilíndrico também está muito próximo (ver Figura 60.a). Pela vista superior do mapa 3D mostrado na Figura 60.(c), relativo à nuvem de pontos densa do RTAB-Map, e pela Figura 60.(d), relativa à nuvem esparsa do Segfloor (segmentada entre *chão* e *não-chão*), tem-se que, ambos mapas 3D apresentam *outliers* nessa região, destacados pelos retângulos tracejados. Isso é causado pela falta de textura, que compromete tanto a técnica ORB quanto a SGM. Para o caso do SGM[2], a Figura 60.(b) mostra o mapa de disparidade com ruídos nessas regiões.

[2]Para o mapeamento da nuvem de pontos, Segfoor utiliza apenas as *features* ORB (disparidade esparsa) e RTAB-Map usa a abordagem SGM (disparidade densa). Segfloor utiliza SGM apenas para segmentar a nuvem de pontos esparsa.

Como nossa técnica só aplica o *ray tracing* com pontos 3D que efetivamente são classificados como "chão", o OGM 2D gerado pelo Segfloor não sofreu impactado negativo com esses *outliers*, os quais foram classificados como "obstáculo". Dessa forma, pela Figura 59.(a), é possível perceber uma coerência visual com relação à delimitação estimada. No caso do RTAB-Map, que gera o *ray tracing* com *voxels* classificados como "obstáculo", as regiões de navegação foram definidas até esses falsos limites representados pelos *outliers*, como mostra a Figura 59.(a/b). Além disso, algumas dessas regiões, na verdade, representam pontos cego para a câmera (atrás ou dentro dos objetos).

Figura 60 – *Outliers* são destacados na nuvem de pontos (c) densa e (d) esparsa, resultantes (a) da baixa textura; (b) ilustra o mapa de disparidade denso com destaque para os ruídos na região não texturizada. Fonte: autora.



Fonte: autora.

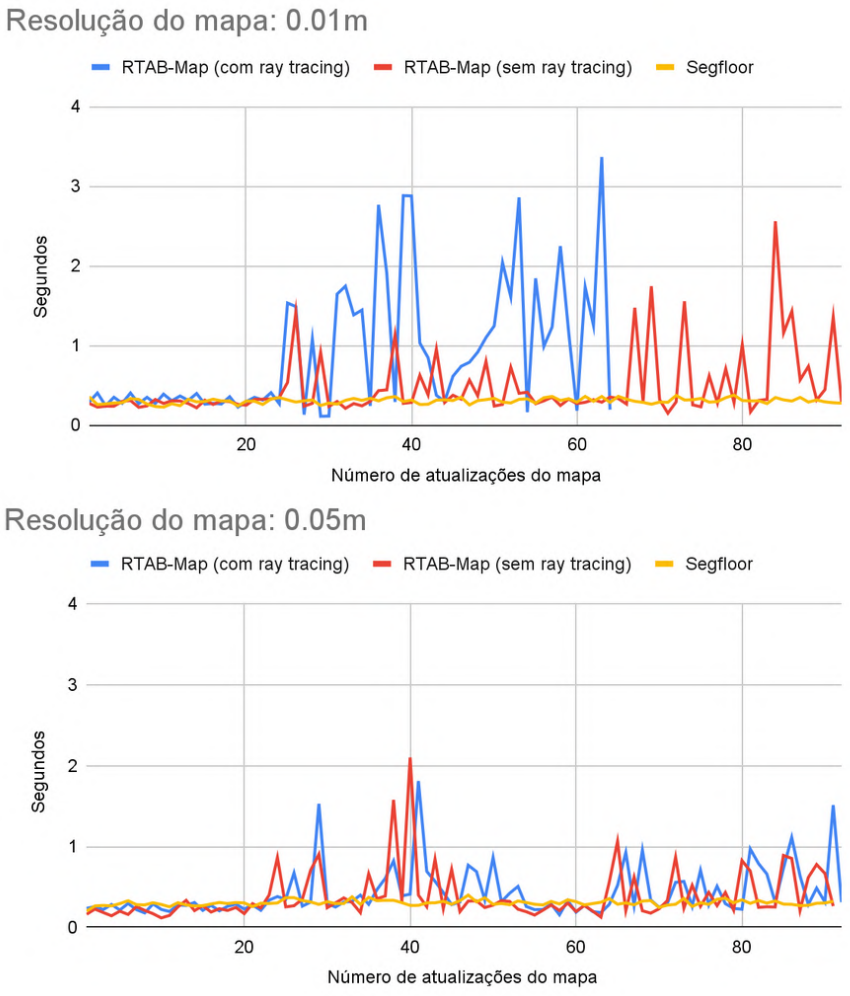
7.1.1 Custo computacional

Para esta cena, foi medido o tempo para a geração do OGM 2D considerando as diferentes configurações de resolução e a opção de *ray tracing*. A medição do tempo para ambas as abordagens começa quando os dados necessários provenientes da técnica SLAM chegam ao respectivo método (ver Figura 51), e termina quando o mapa é gerado e enviado (via ROS). A taxa de atualização do mapa foi definida como $1Hz$, seguindo a recomendação dos autores do RTAB-Map (LABBÉ; MICHAUD, 2019), e essa mesma taxa foi adotada para o Segfloor para fins de comparação. Além do mais, tem-se que nenhum dos programas possui paralelismo na execução medida. Durante a gravação dessa cena, algumas revisitas foram feitas com a finalidade de cobrir bem o ambiente mapeado.

Na Figura 61, os gráficos relativo ao tempo de processamento por atualização do OGM estão separados de acordo com a resolução da grade. No que diz respeito ao Segfloor, nas duas

resoluções, o tempo de processamento foi baixo e constante, com picos de 0.37 e 0.39 segundos nos tamanhos da grade de $0.01m^2$ e $0.05m^2$, respectivamente. Dessa forma, o usuário pode optar por um mapa mais detalhado ou mais seguro de acordo com a sua necessidade, dado que isto não afeta o processamento em tempo real da aplicação. Com relação às revisitas detectadas, tem-se que, como nossa abordagem utiliza o mapeamento esparso, a otimização da nuvem de pontos segmentada é rápida o suficiente para não comprometer a atuação em tempo real da nossa aplicação durante toda sua funcionalidade.

Figura 61 – Tempo de processamento por número de atualizações do mapa. O gráfico superior considera o consumo para gerar mapas com resolução de $0.01m^2$, e o inferior com resolução de $0.05m^2$. Ambos apresentam as curvas relativas ao custo do Segfloor e do RTAB-Map com e sem *ray tracing*.



Fonte: autora.

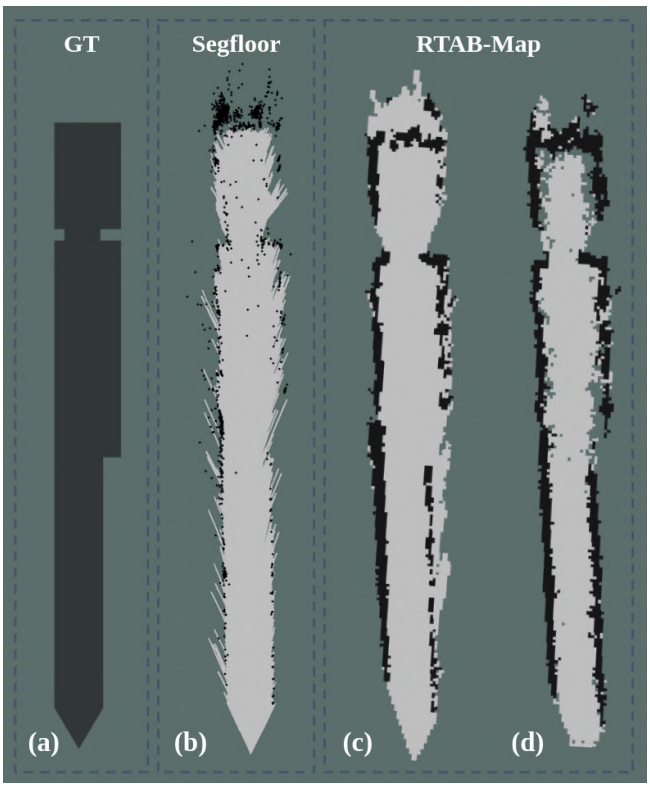
No caso do RTAB-Map, as revisitas detectadas geraram um custo que compromete o seu processamento em tempo real, pois a revisita é o momento que a correção da nuvem de pontos densa acontece, seja de forma global ou local, e que é naturalmente mais custoso (DAI et al., 2017; TAKETOMI; UCHIYAMA; IKEDA, 2017). Outro fator que agrava os custos é a resolução, dado que o RTAB-Map gera primeiro voxels, para depois inferir o estado das grades

de ocupação. Isso quer dizer que quando se diminui a resolução, se aumenta exponencialmente o custo de processamento, pois se trata de um volume cúbico. Para grades de $0.01m^2$, o tempo requerido pelo RTAB-Map apresentou picos de até 3.37 e 2.56 segundos nas opções com e sem *ray tracing*. Na resolução de $0.05m^2$, os picos foram de 1.81 e 2.10 segundos nas opções com e sem *ray tracing*, respectivamente. As demais atualizações foram executadas dentro da taxa de 1 Hz.

7.2 CORREDOR 1 E CORREDOR 2

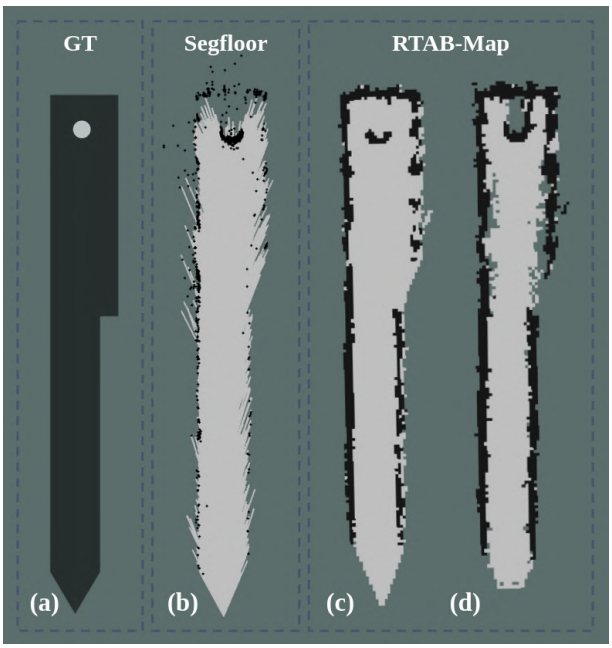
Nesta seção, apresentaremos de forma conjunta dois ambientes semelhantes mas com desafios diferentes (ver Figura 55), nomeados de *corredor 1*, que contém uma passagem semi-aberta; e *corredor 2*, que conta com a presença de um obstáculo cilíndrico. Aqui, adotamos a resolução recomendada de $0.01m^2$ para o Segfloor e a recomendada, de $0.05m^2$, para o RTAB-Map, alterando apenas a opção de mapeamento com ou sem *ray tracing*. Os resultados são ilustrados nas Figuras 62 e 63, onde o *ground truth* também é mostrado. É importante ressaltar que o parâmetro relativo ao comprimento do raio foi definido como, no máximo, 2 metros tanto para o Sefloor como para o RTAB-Map.

Figura 62 – *Corredor 1*: (a) mapa ground truth; (b) OGM gerado pelo Segfloor com resolução de $0.01m^2$; OGM gerado pelo RTAB-Map (c) com e (d) sem *ray tracing*, com resolução de $0.05m^2$.



Fonte: autora.

Figura 63 – *Corredor 2*: (a) mapa ground truth; (b) OGM gerado pelo Segfloor com resolução de $0.01m^2$; OGM gerado pelo RTAB-Map (c) com e (d) sem *ray tracing*, com resolução de $0.05m^2$.



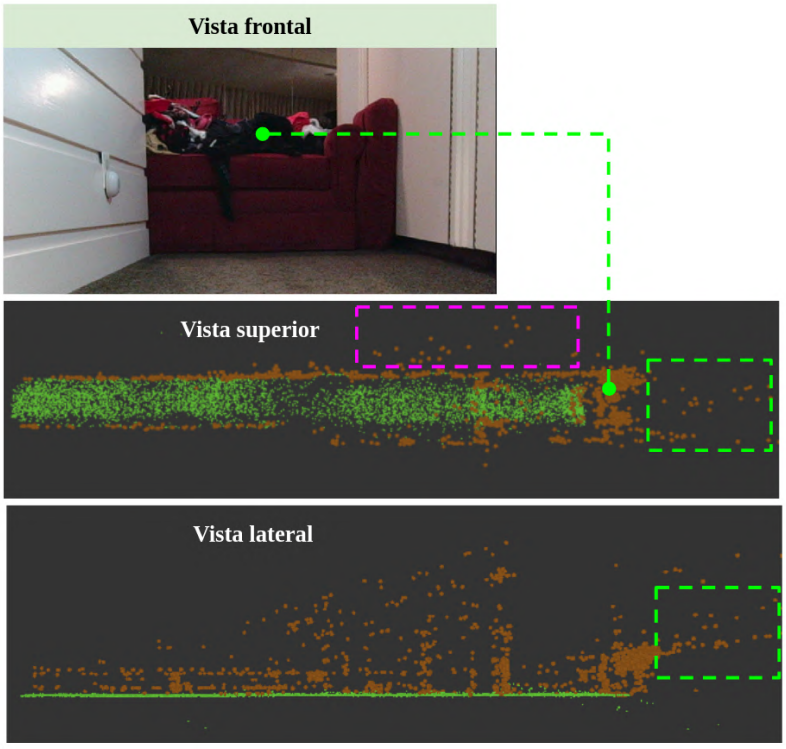
Fonte: autora.

A abordagem proposta, Segfloor, visualmente apresenta uma melhor consistência na definição da área livre para navegação, isto por que só realizamos *ray tracing* com pontos que são classificados como "chão", diferentemente do RTAB-Map que usa os pontos classificados como "osbtáculo" no *ray tracing*. No entanto, um ponto negativo do Segfloor observado para essas cenas é quanto à efetiva percepção das paredes como grades "ocupadas" devido à pouca extração de *features* nessa região, gerando áreas ocupadas de forma muito esparsa.

Na Figura 64, os retângulos tracejados destacam na nuvem de pontos segmentada por nossa técnica, pontos flutuantes no espaço que indicam potencias *outliers*. Como são reportados como "obstáculos", não há *ray tracing* até esses pontos. Apesar deles aparecerem no OGM 2D do Segfloor (Figura 62.b), não representam risco para a navegação do agente, dado que a região livre é delimitada até o sofá, como desejado (ver Figura 62.b). Ainda na Figura 64, o retângulo tracejado em rosa ilustra *outliers* (pontos além da parede) gerados pela baixa textura na parede, onde, mais uma vez, não impactam na geração da região livre.

Quando o *ray tracing* está ativado, o mapa do RTAB-Map tende a ir além da região delimitada, como mostra a Figura 62.(c). Para evidenciar essa problemática, alteramos o limite máximo do raio para 5 metros. A Figura 65 mostra a associação dos objetos na imagem de vista frontal com a nuvem de pontos densa e o OGM 2D, ambos gerados pelo RTAB-Map. Apesar do sofá ser corretamente definido como área ocupada, o *ray tracing* estende a região livre para além do sofá (inclusive dentro dele), até alcançar a máxima delimitação visualizada, neste caso, a cortina, gerando uma falsa área navegável. Esse mesmo fator é percebido na Figura 63.c, onde os raios atravessam o obstáculo cilíndrico, e apontam como área livre as

Figura 64 – Ilustração da nuvem de pontos segmentada pelo Segfloor, onde *outliers* são destacados pelos retângulos tracejados.



Fonte: autora.

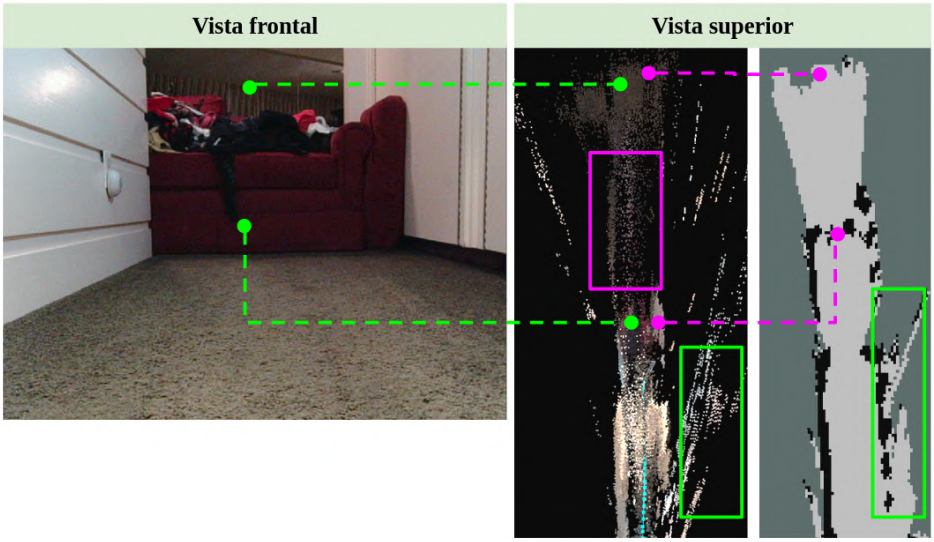
regiões dentro do sofá e atrás dele, que na verdade consiste numa região de ponto cego para a câmera, logo, o mais apropriado seria a indicação de região não definida.

Com relação ao RTAB-Map, a Figura 65 mostra a presença de *outliers* na nuvem de pontos densa e consequentemente no OGM 2D gerado também, destacados pelos retângulos em verde. Esses *outliers* são potencialmente criados devido à baixa textura na parede, impactando a correta associação dos *pixels* correspondentes; e em rosa, possivelmente gerados pela repetição de padrão da cortina e agravado por se tratar de um reflexo no espelho (ambas representam limitações da técnica de disparidade densa). Dessa forma, raios são gerados até esses falsos limites.

Dessa forma, observamos que os ruídos apresentados na nuvem de pontos densa (gerados pela incorreta fusão dos pontos ou pela geração de *outliers*) tendem a levar a maiores impactos negativos para o OGM comparado aos *outliers* na nuvem esparsa, usada pelo Segfloor. Mesmo ambas as abordagens sofrendo com desafios conhecidos da correspondência estéreo, e.g. baixa textura, tem-se que, no mapeamento 3D esparso do sistema SLAM, são mapeados apenas os pontos 3D que apresentam determinada confiabilidade com relação a associação entre *features*. Logo, quando criamos o OGM a partir desses pontos, estamos criando a partir de dados mais confiáveis.

Com relação ao tempo de processamento, a abordagem Segfloor apresentou um valor de pico de 0.39 e 0.38 segundos para o corredor 1 e 2, respectivamente. Vale relembrar que as

Figura 65 – Ilustração da problemática relacionada ao *ray tracing* quando este leva em consideração grades definidas como "obstáculo" em vez de "livre". Área navegável é indicada além de delimitações como a parede e o sofá.



Fonte: autora.

otimizações da nuvem de pontos do sistema SLAM são replicadas na nuvem segmentada do nosso módulo em cada atualização do OGM. Por parte do RTAB-Map, houve uma correção da nuvem de pontos densa local que levou 1.23 segundos. Para o *corredor 2*, ocorreram quatro otimizações locais, com custo máximo de 2.70 segundos e os demais entre 1.11 e 1.27 segundos.

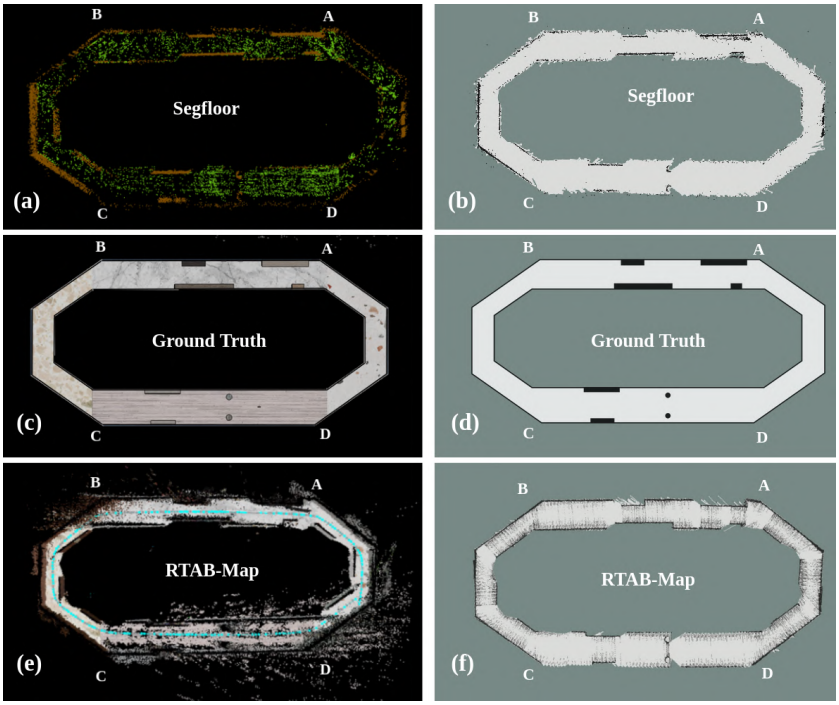
Com relação às cenas apresentadas nessa seção, notamos que todos os OGMs gerados foram visualmente satisfatórios, especialmente quando utilizamos nossa abordagem ou a do RTAB-Map na opção sem *ray tracing*. No entanto, com relação ao tempo de processamento, foi evidenciado uma vantagem com o uso do Segfloor, dado que as otimizações da nuvem esparsa não comprometeram a atuação *online* da proposta. Ainda mais, é importante lembrar que o Segfloor mapeou em uma resolução cinco vezes maior. A interpretação quantitativa, através da precisão e *recall* em conjunto com a sobreposição dos mapas, é um tópico abordado na Seção 7.4.

7.3 CENA SINTÉTICA

A cena nomeada de *sintética* corresponde a uma área mapeada de $682.7m^2$, com um total de 1007 *keyframes*. Apesar de avaliarmos os mapas gerados confrontando-os com o *ground truth*, nossa principal intenção com esse ambiente foi de obter métricas relativas ao custo computacional e uso de memória para ambas as abordagens Segfloor e RTAB-Map diante de um extenso mapeamento. Portanto, apresentaremos nessa seção uma avaliação qualitativa dos mapas e os custos relativos à geração do OGM 2D a partir da nuvem de pontos densa (RTAB-Map) ou esparsa (Segfloor).

A Figura 66, mostra a vista superior do ambiente sintético criado, além das nuvens de pontos geradas pelas abordagens avaliadas neste trabalho. Com relação à nuvem de pontos segmentada pela nossa abordagem, mostrada na Figura 66.(a), é possível perceber baixa quantidade de *outliers*. A nuvem esparsa indicada como "chão" (em verde) é visualmente coerente, e consequentemente, a região navegável do OGM equivalente, também. A delimitação da cena foi indicada como "obstáculo" de forma satisfatória onde as paredes apresentam textura (ver Figura 56), isto é, no segmento BC, e parcialmente nos demais segmentos. Demais semelhanças serão evidenciados na sobreposição do mapa e discutidos na Seção 7.4.

Figura 66 – Vista superior da cena sintética; da nuvem de pontos segmentada (Segfloor); da nuvem de pontos densa (RTAB-Map); e o respectivo mapa OGM 2D.



Fonte: autora.

O mapa 3D do RTAB-Map, mostrado na Figura 66.(e), apresenta coerência na maior parte da trajetória do segmento ABC. No entanto, para o segmento CDA há bastante pontos 3D ruidosos, cuja fusão dos pontos 3D correspondentes falhou, gerando diversos *outliers*. Por conta do limite máximo do *ray tracing* (para esta cena definido como 5 metros), o OGM 2D mostrado na Figura 66.(f) para esses segmentos no OGM - Figura 66.(f), não chega até esses falsos limites. A delimitação da cena é indicada de forma satisfatória e para todo o trajeto, no entanto, um pouco menos denso no trecho CD devido aos ruídos excessivo nessa região. Foi observado também que o OGM apresenta um estriado preto durante todo o trajeto, porém, não conseguimos identificar a causa dessa falha.

7.3.1 Métricas Computacionais

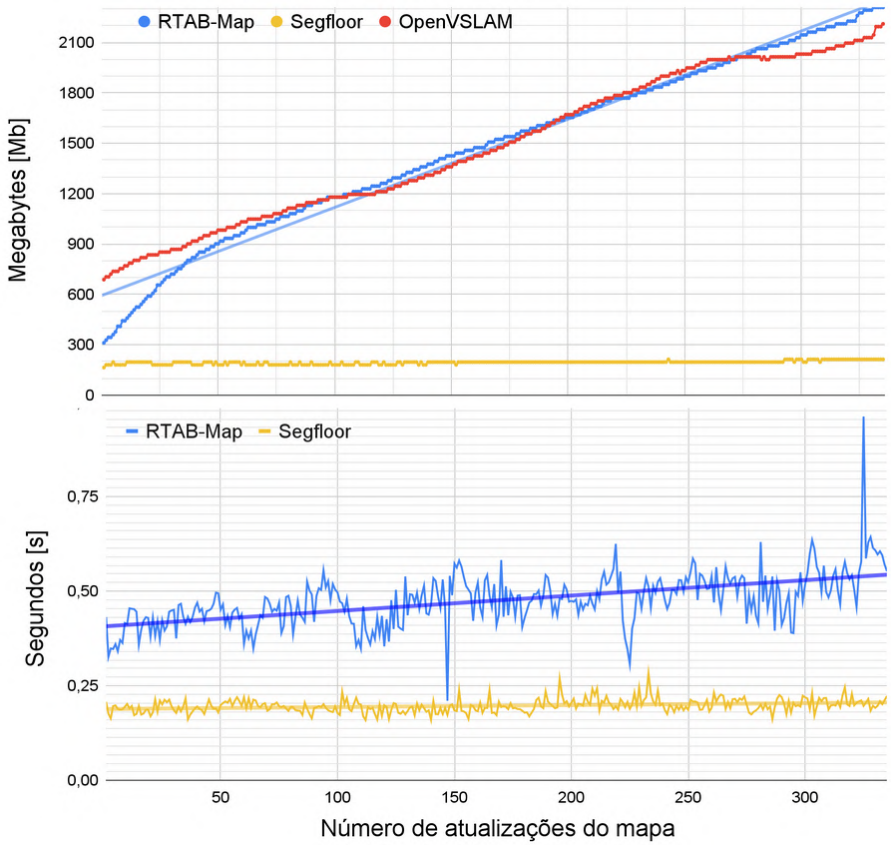
Para avaliar o tempo de processamento e o consumo de memória das abordagens em análise nesse extenso mapeamento, adotamos a taxa de atualização do mapa de $1Hz$ para o RTAB-Map, conforme sugerido pelos autores (LABBÉ; MICHAUD, 2019). Dessa forma, estendemos o uso dessa taxa em nossa aplicação. Como mencionado na Seção 6, Figura 51, o RTAB-Map está rodando em uma configuração de odometria externa (OpenVSLAM). A medição do tempo para ambas as abordagens começa quando os dados necessários provenientes da técnica SLAM chegam ao respectivo método e termina quando o mapa é gerado e enviado (via ROS2). É importante mencionar que nenhum dos programas está sendo executado com paralelismo de *threads* na execução medida.

O gráfico superior da Figura 67 traz o consumo de memória sobre o número de atualizações do mapa. É importante relembrar que tanto o Segfloor quanto o RTAB-Map fazerem uso de informações provenientes do sistema SLAM (ver Figura 51), ou seja, é um sistema comum para ambas as abordagens. Por isso, três curvas são mostradas, referentes ao consumo do Segfloor, do RTAB-Map e do OpenVSLAM. Em cada curva, destacamos uma linha de tendência para projeções de longo prazo. Desta linha, pode-se citar que o RTAB-Map tem uma taxa de crescimento de consumo de memória 71.62 vezes maior que o Segfloor, que tem uma taxa de crescimento de 0.073. É interessante mencionar que usar o OpenvSLAM associado ao RTAB-Map é equivalente a ter dois algoritmos SLAM rodando em termos de consumo de memória. Por outro lado, usar o OpenvSLAM em conjunto com o Segfloor resulta num pequeno acréscimo do consumo de memória para o sistema SLAM.

O gráfico inferior da Figura 67, ilustra o tempo de processamento em função de atualizações do mapa. Sobre os dados brutos foi adicionada uma linha de tendência que destaca um comportamento quase constante da abordagem proposta em comparação ao custo crescente do RTAB-Map. Para ser preciso, o tempo de processamento inicial do RTAB-Map é cerca de duas vezes o tempo do Segfloor, e essa diferença tende a aumentar. Por exemplo, quando a aplicação atinge 300 atualizações, a diferença passa a ser 2.35 vezes maior. Além disso, é possível identificar um pico de 0.96 segundos no final da aplicação do RTAB-Map, relativo ao custo de otimização do mapa denso. No sistema proposto, as otimizações da nuvem de pontos (realizadas no módulo do sistema SLAM e replicadas no módulo de segmentação da nuvem de pontos) acontecem em cada atualização do OGM, no entanto, o tempo de processamento máximo foi de $0,.8$ segundos.

O tempo médio de processamento do Segfloor por atualização de mapa foi de 0.19 segundos, enquanto o tempo médio do RTAB-Map foi de 0.47 segundos, o que significa que o sistema proposto, Segfloor, estaria pronto para fornecer uma taxa de atualização do mapa de até 5Hz, onde uma das principais vantagens de uma taxa mais rápida é fornecer um maior tempo de resposta para o agente atuar.

Figura 67 – Métricas computacionais: consumo de memória em *megabytes* e tempo de processamento em segundos, ambos sobre o número de atualizações do mapa.



Fonte: autora.

7.4 SOBREPOSIÇÃO DE MAPAS, PRECISÃO E COBERTURA

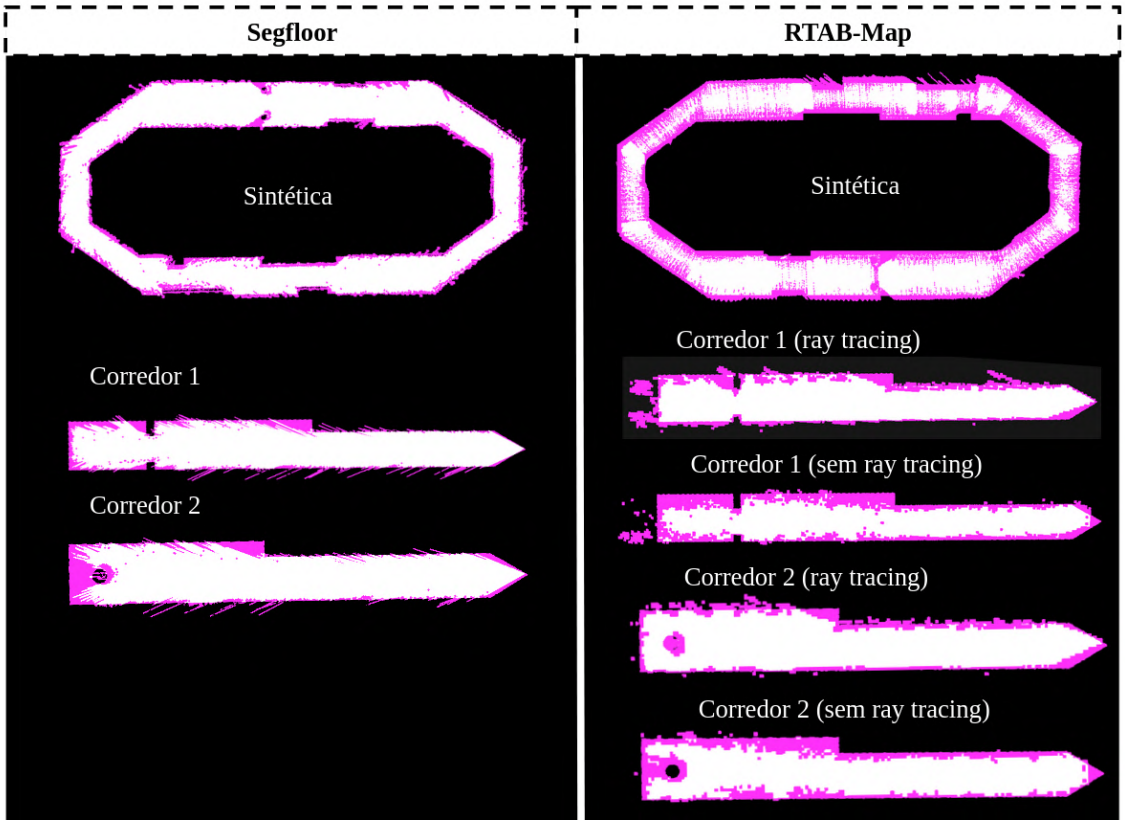
Para as cenas nomeadas de *corredor 1*, *corredor 2* e *sintética*, foram extraídos os mapas *ground truth* (GT). As sobreposições de cada um deles com o respectivo OGM gerado por ambas as estratégias, Segfloor e RTAB-Map, são mostradas na Figura 68. Os mapas foram salvos em formato apropriado (.pgm), e alinhados manualmente (translação e rotação), porém, sem alteração na escala, dado que os mapas já são gerados em tamanho real. Para cada mapa, a região destacada em rosa representa incompatibilidade do estado da grade com relação ao *ground truth*; a cor branca indica quando os dois mapas coincidem suas respectivas regiões navegáveis e preta quando coincidem as regiões definidas como ocupadas. Portanto, quanto menos aparecer a coloração rosa, melhor a qualidade do mapa gerado.

Com relação aos corredores 1 e 2 e a cena sintética, é possível pontuar que o Segfloor conseguiu perceber e replicar a mudança de espessura do trajeto como desejado. Na cena sintética, as pilastras também foram indicadas como grades ocupadas. Para o corredor 1, a passagem relativa à porta semi-aberta também foi indicada, e para o corredor 2 a detecção do obstáculo cilíndrico foi definida como ocupada, onde boa parte da região rosa, de incompatibilidade, se dá em pontos cego da câmera. De uma forma geral, para os três mapas gerados

pelo Segfloor, há bastante coincidência quanto à região navegável (coloração branca), onde poucos raios extrapolaram a delimitação da parede, mas de maneira esparsa, sem chegar a formar uma efetiva região de navegação.

Os mapas gerados com o RTAB-Map também retratam a mudança de espessura durante o trajeto, mais evidenciado nos mapas relativos aos corredores 1 e 2 na opção com *ray tracing*, e na cena sintética. Uma maior região de coloração rosa é percebida nos mapas obtidos sem *ray tracing* e, principalmente, na cena sintética devido ao estiramento de áreas ocupadas por todo o trajeto.

Figura 68 – Sobreposição dos mapas gerados com o respectivo mapa *ground truth*. Coloração rosa indica incompatibilidade: quanto menos aparecer, melhor.



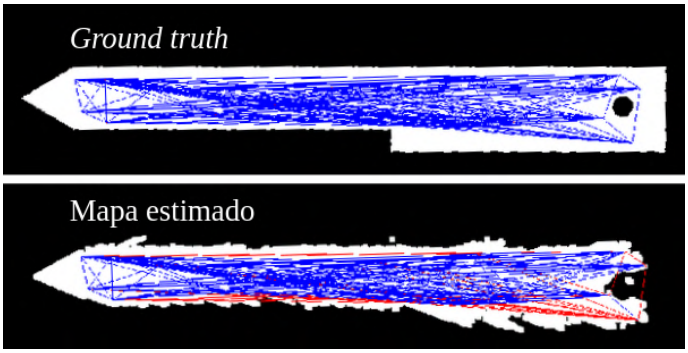
Fonte: autora.

Para quantificar a qualidade dos mapas obtidos para navegação, utilizamos um mapa de rotas probabilístico (PRM), como descrito na Seção 6.1.1. Métricas de precisão são calculadas a partir do PRM no mapa estimado, e replicando-o no *ground truth*, onde os caminhos que induziriam a uma colisão representam caminhos falsos-positivo. Métricas de *recall* são calculadas a partir do PRM no *ground truth*, e replicando-o no mapa estimado. Seu valor indica a porcentagem de dados classificados como "positivo" comparado com a quantidade real de todos os caminhos positivos definidos no *ground truth*.

Apenas para fins ilustrativos, a Figura 69 mostra um desses grafos gerados, onde foram usados 20 nós, alocados de forma aleatória na região do mapa gerado pelo Segfloor. Trajetos

replicados com sucesso no mapa estimado são representados em azul, e os que induziriam a uma colisão, em vermelho.

Figura 69 – Ilustração do PRM.



Fonte: autora.

A fim de mitigar equivocadas classificações das grades de ocupação, aplicamos um pós-processamento nos OGMs gerados pelas duas abordagens. Partimos do pré-suposto que uma região definida como "obstáculo" que se apresente isolada dentro da região navegável indica um comportamento atípico, logo, é um potencial *outlier*. No caso do Segfloor, observamos alguns pontos 3D da nuvem esparsa flutuando no espaço livre, provável *outlier*, onde sua projetação 2D reproduziu o erro no OGM. Com relação aos erros nos mapas gerados pelo RTAB-Map, a causa mais comum de reprodução de erro foi devido ao *ray tracing* até falsos limites (ver Figura 65).

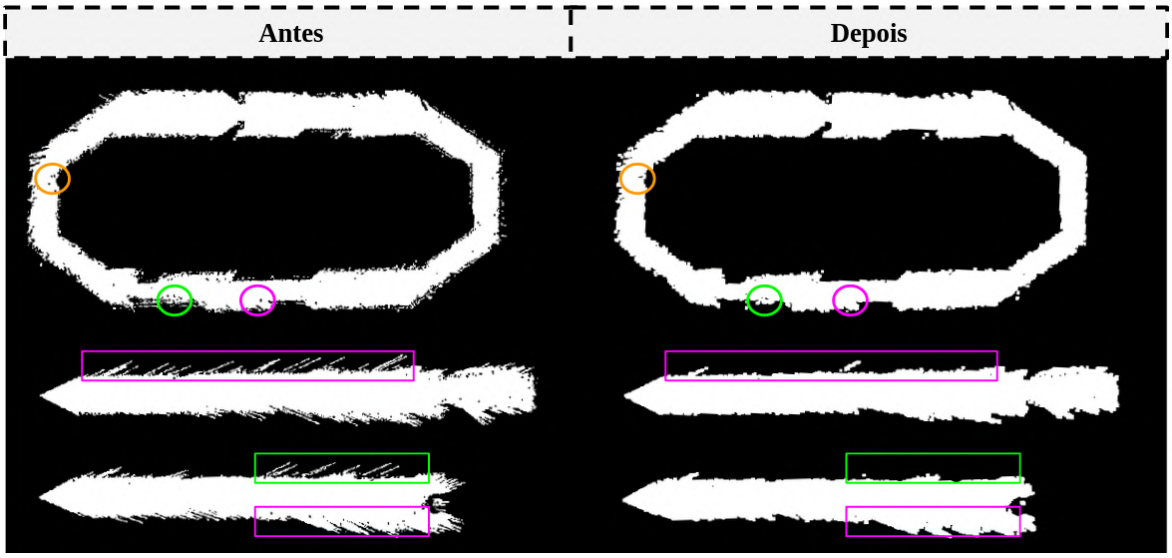
Adotamos um processamento de imagem chamado de *erosão*, seguido de *dilatação*, que consistem em operações morfológicas (SOILLE, 2004; CYGANEK; SIEBERT, 2011). Para o nosso caso, a erosão é capaz de remover os raios e pontos isolados de forma que a região livre seja pouco impactada. Em seguida, a dilatação é utilizada para que a região navegável que, eventualmente, tenha diminuído com a erosão, volte ao tamanho original.

A Figura 70 ilustra os resultados para os mapas gerados pelo Segfloor (antes e depois). Está destacado a mitigação de raios esparsos e pontos isolados na região livre. É interessante ressaltar que tal estratégia também foi aplicado nos mapas do RTAB-Map antes de levantar os dados de precisão e *recall*, assim, seus mapas foram igualmente aprimorados.

A Tabela 3 traz maiores detalhes sobre cada cena mapeada cujo mapa *ground truth* foi extraído. É informado o tamanho das cenas em metros quadrados; o número de *keyframes* gerados pelo sistema SLAM; e o número de nós do PRM, definidos de forma que representassem uma boa cobertura da região mapeada. Como os nós do grafo de rotas são alocados de forma aleatória, rodamos a avaliação 30 vezes, e o resultado apresentado é, na verdade, a média dessa repetição.

A Tabela 4 traz os resultados relativos à precisão, calculados considerando a ferramenta de planejamento de rotas, o PRM. A precisão indica quantos caminhos, dentre as trajetórias estimadas no mapa gerado, levariam a uma colisão. Como se pode perceber pela Figura 68,

Figura 70 – Antes e depois do pós-processamento para tratamento de pontos e raios *outliers*. Os retângulos e círculos destacam exemplos de melhorias.



Fonte: autora.

Tabela 3 – Cenas mapeadas e seus tamanhos em metros quadrados; o número de *keyframes* gerados pelo sistema SLAM; e o número de nós do PRM.

Cena	Sintética	Corredor 1	Corredor 2
Tamanho	682.7m^2	10.0m^2	7.9m^2
Número de KFs	1007	354	220
Número de nós	100	20	20

Fonte: autora.

a maioria da região livre estimada coincide com alguma região livre do *ground truth*. Dessa forma, as duas abordagens apresentam altíssimos valores de precisão, ou seja, quase nenhum trajeto do PRM no OGM gerado pelas duas estratégias resultaria em uma colisão.

Tabela 4 – Precisão dos mapas gerados.

	Sintética	Corredor 1	Corredor 2
Segfloor	99,41%	99.48%	99.64%
RTAB-Map - ray tracing	98.44%	96.63%	99.22%
RTAB-Map - sem ray tracing	-	99.66%	99.31%

Fonte: autora.

No entanto, quanto dessa área livre estimada representa do total de região livre do *ground truth*? Quanto dessa região livre total o robô acessaria/navegaria de acordo com *seu* mapeamento? A métrica *recall* foi utilizada para inferir a porcentagem da cobertura de cada estratégia, mostrada na Tabela 5. Os mapas gerados pelo Segfloor apresentaram as maiores porcentagens com relação ao mapeamento da área livre total da cena. Associando

as porcentagens com a Figura 68, se percebe a coerência desse resultado, onde se observa pouca coloração rosa na região que diz respeito à área livre do *ground truth*. Para o caso da cena sintética, por exemplo, tem-se que 98.52% dos caminhos do PRM gerados do *ground truth* conseguiriam ser replicados com sucesso no mapa estimado pelo Segfloor. Dessa forma, as métricas de precisão e *recall* indicam que os mapas gerados por nossa técnica dificilmente induziria a uma colisão e, adicionalmente, conseguiria cobrir bem a região livre disponível para a navegação.

Tabela 5 – *Recall* (cobertura) dos mapas gerados.

	Sintética	Corredor 1	Corredor 2
Segfloor	**98,52%**	**97.70%**	**97,23%**
RTAB-Map - ray tracing	31,67%	94.94%	96.21%
RTAB-Map - sem ray tracing	-	59.06%	71.73%

Fonte: autora.

Como esperado, o mapeamento do ambiente sintético por parte do RTAB-Map registrou pouca área livre comparado ao total de área navegável na cena, mais especificamente, 31.67%. O mesmo se percebe para o corredor 1 e 2 sem *ray tracing*, com uma cobertura de 59.05% e 71.73%, respectivamente. Com a opção de *ray tracing* ativada, os mapas gerados pelo RTAB-Map relativos aos corredores 1 e 2 apresentaram resultados bastante competitivos com a proposta Segfloor.

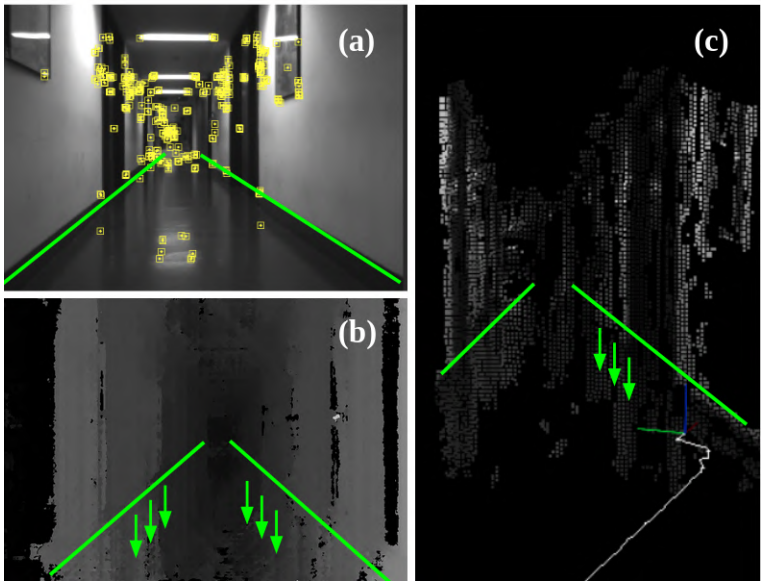
7.5 LIMITAÇÃO

Como brevemente abordado no Capítulo 1, a técnica apresentada nesta dissertação tem como pré-requisito que o ambiente a ser mapeado apresente regiões com textura, especialmente na área navegável. Isto porque a falta de textura compromete a extração de *features* e mapeamento por parte do sistema SLAM numa região de interesse, além de comprometer o mapa de disparidade denso relativo à técnica de segmentação. É importante frisar que a textura contém elementos visuais cujas detecções acorrem como descrito na Seção 3.3.1. Dessa forma, um ambiente com um chão majoritariamente liso, representa uma limitação deste trabalho.

Essa problemática é ilustrada com o auxílio da base de dados pública *Rawseeds*, onde a maioria dos trechos eram de cor preto-opaco, e um pequeno trecho continha chão com textura, isto é, estava dentro do pré-requisito da proposta. A Figura 71.(a) ilustra um trecho com chão preto e opaco, refletindo a luz do teto. Apesar do sistema SLAM conseguir extrair *features* suficientes (pontos em amarelo) para se localizar, é possível observar pouquíssimas *features* na região livre, o chão. A Figura 71.(b) traz o mapa de disparidade denso gerado pelo algoritmo de correspondência estéreo SGM; (c) a nuvem de pontos formada pelo algoritmo do RTAB-Map no mesmo corredor. Em todas as figuras está destacado o perfil do chão com segmentos em verde. É possível perceber que o mapa de disparidade não condiz com o que se

observa do corredor, pois a parede aparece com continuidade no sentido indicado pelas setas em verde.

Figura 71 – Ilustração (a) de um trecho navegado, cujo chão é opaco e preto. Pontos em amarelo indicam *features* extraídas; (b) o mapa de disparidade para (a), onde setas indicam a continuidade equivocada da parede; (c) a nuvem de pontos densa gerada pelo RTAB-Map que reproduz o erro na estimativa da profundidade nos pontos 3D.



Fonte: autora.

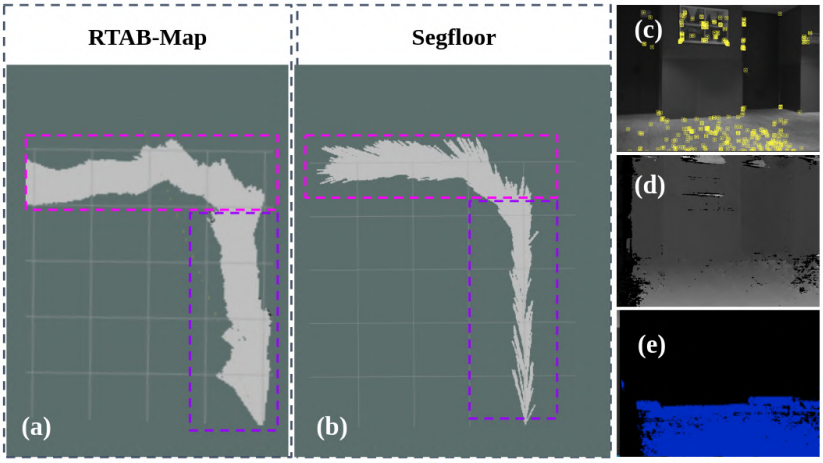
Como detalhado na Seção 2.5.3, quando um algoritmo como o SGM, de abordagem semi-global, não encontra os *pixels* correspondentes (por exemplo, devido a falta de textura), ele busca determinar a disparidade pelo valor que resulte no menor fator de energia de suavização (Equação 2.20). Isso parte do pressuposto que uma superfície não apresenta descontinuidades bruscas, e valores de disparidade que suavizem o resultado são favorecidos, que é o que se observa na Figura 71.(b).

O mapa de disparidade gerado pelo SGM é tanto usado pela nossa proposta quanto pela do RTAB-Map, logo, este caso compromete ambos os sistemas. A nuvem de pontos gerada, mostrada na Figura 71.(c), traz a reprodução do erro na estimativa de profundidade dos pontos 3D, ficando nítido que não há região de chão indicada. Com a opção de *ray tracing* ativada, o RTAB-Map consegue lidar com isso, não por detectar o chão, mas por detectar os obstáculos e assumir que estes indicam o limite da área livre.

A Figura 72 compara o mapeamento realizado por ambas as estratégias, onde os retângulos tracejados em roxo destacam o trecho discutido acima, isto é, sem textura no chão. Para este trajeto, fica evidenciado na Figura 72.(b) um mapa menos denso, com linhas esparsas, resultante de *outliers*, gerados pela falha do SGM e, consequentemente, da segmentação. Em contra partida, o trecho destacado pelo retângulo tracejado em rosa, apresenta região com textura no chão, ilustrado pela Figura 72.(c), que também destaca as *features*, as quais foram

suficientes para garantir uma boa cobertura da área livre. Além disso, o mapa de disparidade em (d) reproduz corretamente o que se espera, e consequentemente, a segmentação em (e) também tem êxito. As duas abordagens, RTAB-Map e Segfloor, apresentam resultados bem semelhantes nessa região.

Figura 72 – Mapas gerados por ambas abordagens. A região destacada pelos retângulos em (a) e (b) dividem a região mapeada baseado na ausência ou presenta de textura no chão. (c) traz uma imagem do trecho com chão texturizado e as respectivas *features* extraídas; (d) e (e) mostram a disparidade e a segmentação, respetivamente, para essa imagem.



Fonte: autora.

8 CONCLUSÃO

A técnica proposta nesta dissertação atua como extensão da etapa de mapeamento de um sistema vSLAM baseado em *features*. A partir da nuvem de pontos esparsa gerada pelo vSLAM, em conjunto com uma técnica de segmentação de imagem, foi possível gerar um OGM 2D. Além do mapa, o sistema proposto também gera a nuvem de pontos esparsa segmentada entre "chão" e "não-chao", a qual é atualizada para cada novo *keyframe* (momento que o algoritmo SLAM cria e remove pontos 3D da nuvem).

O método proposto representa um elo entre métodos vSLAM de mapeamento esparso e atividades de navegação, auxiliando o agente quanto à percepção da cena através da geração de um tipo de mapa amplamente utilizada em aplicações robóticas. A técnica proposta para a geração do OGM acrescenta ao processo SLAM um baixo consumo de memória e atua sem comprometer o processo *online* da aplicação. Além disso, o sistema desenvolvido requer apenas uma câmera estéreo, sendo uma solução de menor investimento financeiro comparado com demais alternativas.

Para testar a efetividade do método proposto, coletamos dados de cenas em ambiente real, onde cada uma apresentou um tipo diferente de desafio, como a presença de obstáculos e de vários elementos de tamanhos, formas e texturas diferentes. Além disso, também foi utilizada uma cena desenvolvida em ambiente sintético, com desafios de baixa textura, obstáculos, prateleiras e variação de largura do corredor. A cena sintética foi essencial para avaliar o comportamento da técnica proposta frente à um mapeamento extenso dado que ela possui uma trajetória de 138.4 metros.

Comparamos os mapas gerados pelo método proposto, Segfloor, com os gerados pelo projeto de código aberto RTAM-Map (LABBÉ; MICHAUD, 2019). Visualmente, ambas as estratégias apresentaram mapas similares, com boa cobertura da região navegável, onde a principal diferença observada foi com relação à tolerância da técnica para *outliers* proveniente da nuvem de pontos 3D. Segfloor se mostrou mais robusto na presença de *outliers*, isto porque a técnica de *ray tracing* é executada considerando apenas pontos que foram classificados como "chão", que por estarem mais próximos do agente, naturalmente apresentam mapeamento mais preciso e com menos ruídos nessa região.

Através da sobreposição dos mapas gerados pelo Segfloor com seu respectivo *ground truth*, ficou evidenciado uma boa interseção entre as regiões livres. A comprovação numérica se deu por meio de um grafo de trajetórias (PRM) sobre o mapa gerado, alcançando resultados de alta precisão de 99% e alta cobertura (*recall*), de 97% ou mais.

Na avaliação de métricas computacionais, ficou comprovado que a proposta conseguiu atuar como um módulo de baixo consumo de memória e de rápido processamento, garantindo uma alternativa que funciona *online* junto ao algoritmo SLAM. A técnica atinge um tempo de processamento que permite a atualização do mapa em $5Hz$. Em contra-partida, o RTAB-Map,

que passa por uma etapa de mapeamento denso antes de gerar o OGM 2D, apresentou um crescimento linear quanto ao consumo de memória (equivalente ao do sistema SLAM), e com relação ao tempo de processamento, ficou evidenciado quão custoso pode ser o mapeamento 2D através da nuvem densa, onde a otimização dos pontos 3D resultou em um custo de mais de 3 segundos na atualização do OGM 2D.

Em suma, o trabalho alcançou as finalidades propostas, conforme apresentadas no Capítulo 1, são elas:

- gerar o OGM a partir da associação do estágio de mapeamento do vSLAM com uma técnica de segmentação de imagem;
- representar um módulo complementar que transforma a nuvem de pontos esparsa em um OGM 2D e também numa nuvem segmentada entre *chão* e *não-chão*;
- ser um módulo que leva à considerável redução do uso de memória mesmo para uma cena de larga escala.

A limitação da abordagem proposta consiste em cenas cuja região navegável não apresente textura, conforme discutido e apresentado a partir da base de dados pública Rawseeds (CERIANI et al., 2009). Devido a falta de textura, poucas *features* são extraídas na região de interesse, comprometendo a densidade dos raios gerados pelo *ray tracing*. Um segundo fator mais agravante, é que a baixa textura é uma limitação da técnica de correspondência estéreo densa, comprometendo diretamente a correta segmentação da imagem.

Intenções de trabalhos futuros consistem na integração do sensor IMU como forma de aumentar a precisão da estimativa de localização do sistema SLAM, como evidenciado pelo trabalho descrito em (LING; SHEN, 2019). Além disso, por se tratar de uma proposta de baixo custo computacional, espera-se diminuir ainda mais esse custo embarcando a técnica em FPGA de forma a associá-la com o trabalho desenvolvido em (CAMBUIM, 2017), do cálculo de disparidade densa em FPGA, e do trabalho (ISHIMARU, 2021), de segmentação de imagem (chão), também embarcada em FPGA. Adicionalmente, espera-se avaliar a possibilidade de técnicas que executem uma correspondência estéreo mais precisa para ambientes de baixa textura. Também representa uma intenção futura estender o mapa 2D binário para um contexto probabilístico, de forma a garantir uma percepção mais segura da cena, além de ser uma representação mais adequada para a fusão de mapas gerados com outros sensores, caso integrados. Por fim, a adesão de informação semântica no mapa 2D é totalmente viável a partir da câmera como exemplificado pelo trabalho descrito em (QI et al., 2020).

REFERÊNCIAS

AZZAM, R.; TAHA, T.; HUANG, S.; ZWEIRI, Y. Feature-based visual simultaneous localization and mapping: a survey. *SN Applied Sciences*, Springer, v. 2, n. 2, p. 1–24, 2020.

BALAGUER, B.; BALAKIRSKY, S.; CARPIN, S.; VISSER, A. Evaluating maps produced by urban search and rescue robots: lessons learned from robocup. *Autonomous Robots*, Springer, v. 27, n. 4, p. 449–464, 2009.

BLEYER, M.; BREITENEDER, C. Stereo matching—state-of-the-art and research challenges. In: *Advanced topics in computer vision*. [S.l.]: Springer, 2013. p. 143–179.

BLOCHLIGER, F.; FEHR, M.; DYMCZYK, M.; SCHNEIDER, T.; SIEGWART, R. Topomap: Topological mapping and navigation based on visual slam maps. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 3818–3825.

BORGHESE, N. A.; CERVERI, P. Calibrating a video camera pair with a rigid bar. *Pattern Recognition*, Elsevier, v. 33, n. 1, p. 81–95, 2000.

BOUGUET, J.-Y. *A few links related to camera calibration*. 2015. <http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/links.html>. [Online; Acessado 20-Setembro-2021].

BROGGI, A.; CARDARELLI, E.; CATTANI, S.; SABBATELLI, M. Terrain mapping for off-road autonomous ground vehicles using rational b-spline surfaces and stereo vision. In: IEEE. *2013 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2013. p. 648–653.

BUFFA, M.; FAUGERAS, O. D.; ZHANG, Z. Obstacle avoidance and trajectory planning for an indoor mobile robot using stereo vision and delaunay triangulation. In: *Vision-based Vehicle Guidance*. [S.l.]: Springer, 1992. p. 268–283.

CADENA, C.; CARLONE, L.; CARRILLO, H.; LATIF, Y.; SCARAMUZZA, D.; NEIRA, J.; REID, I.; LEONARD, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, IEEE, v. 32, n. 6, p. 1309–1332, 2016.

CALONDER, M.; LEPETIT, V.; STRECHA, C.; FUA, P. Brief: Binary robust independent elementary features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2010. p. 778–792.

CAMBUIM, L. F. d. S. *Um módulo de hardware de tempo real de correspondência semi global para um sistema de visão estéreo*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017.

CAMPOS, C.; ELVIRA, R.; RODRÍGUEZ, J. J. G.; MONTIEL, J. M.; TARDÓS, J. D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, IEEE, 2021.

CERIANI, S.; FONTANA, G.; GIUSTI, A.; MARZORATI, D.; MATTEUCCI, M.; MIGLIORE, D.; RIZZI, D.; SORRENTI, D. G.; TADDEI, P. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, Springer, v. 27, n. 4, p. 353–371, 2009.

CHEN, L.-C.; ZHU, Y.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 801–818.

CHEN, Z.; LIU, L. Navigable space construction from sparse noisy point clouds. *IEEE Robotics and Automation Letters*, IEEE, v. 6, n. 3, p. 4720–4727, 2021.

COLLINS, T.; COLLINS, J. Occupancy grid mapping: An empirical evaluation. In: IEEE. *2007 mediterranean conference on control & automation*. [S.l.], 2007. p. 1–6.

CURLESS, B.; LEVOY, M. A volumetric method for building complex models from range images. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1996. p. 303–312.

CYGANEK, B.; SIEBERT, J. P. *An introduction to 3D computer vision techniques and algorithms*. [S.l.]: John Wiley & Sons, 2011.

DAI, A.; NIESSNER, M.; ZOLLHÖFER, M.; IZADI, S.; THEOBALT, C. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, ACM New York, NY, USA, v. 36, n. 4, p. 1, 2017.

DAVISON, A. J.; REID, I. D.; MOLTON, N. D.; STASSE, O. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 29, n. 6, p. 1052–1067, 2007.

DEITS, R.; TEDRAKE, R. Computing large convex regions of obstacle-free space through semidefinite programming. In: *Algorithmic foundations of robotics XI*. [S.l.]: Springer, 2015. p. 109–124.

DUBERG, D.; JENSFELT, P. Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 4, p. 6411–6418, 2020.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, IEEE, v. 22, n. 6, p. 46–57, 1989.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 834–849.

ESRAFILIAN, O. *Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM*. 2016. <https://www.youtube.com/watch?v=vIR7NgsMt6U>. [Online; Acessado 20-Setembro-2021].

ESRAFILIAN, O.; TAGHIRAD, H. D. Autonomous flight and obstacle avoidance of a quadrotor by monocular slam. In: IEEE. *2016 4th International Conference on Robotics and Mechatronics (ICROM)*. [S.l.], 2016. p. 240–245.

FALLON, M.; JOHANNSSON, H.; KAESS, M.; LEONARD, J. J. The mit stata center dataset. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 32, n. 14, p. 1695–1699, 2013.

FAN, R.; WANG, L.; BOCUS, M. J.; PITAS, I. Computer stereo vision for autonomous driving. *arXiv preprint arXiv:2012.03194*, 2020.

FILIPENKO, M.; AFANASYEV, I. Comparison of various slam systems for mobile robot in an indoor environment. In: IEEE. *2018 International Conference on Intelligent Systems (IS)*. [S.l.], 2018. p. 400–407.

FILLIAT, D. A visual bag of words method for interactive qualitative localization and mapping. In: IEEE. *Proceedings 2007 IEEE International Conference on Robotics and Automation*. [S.l.], 2007. p. 3921–3926.

GAKNE, P. V.; O'KEEFE, K. Tightly-coupled gnss/vision using a sky-pointing camera for vehicle navigation in urban areas. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 4, p. 1244, 2018.

GÁLVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, IEEE, v. 28, n. 5, p. 1188–1197, 2012.

GAO, X.; WANG, R.; DEMMEL, N.; CREMERS, D. Ldso: Direct sparse odometry with loop closure. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2018. p. 2198–2204.

GEIGER, A.; ROSER, M.; URTASUN, R. Efficient large-scale stereo matching. In: SPRINGER. *Asian conference on computer vision*. [S.l.], 2010. p. 25–38.

GOMEZ-OJEDA, R.; MORENO, F.-A.; ZUNIGA-NOËL, D.; SCARAMUZZA, D.; GONZALEZ-JIMENEZ, J. Pl-slam: A stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics*, IEEE, v. 35, n. 3, p. 734–746, 2019.

HAMZAH, R. A.; IBRAHIM, H. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, Hindawi, v. 2016, 2016.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2. ed. [S.l.]: Cambridge University Press, 2004.

HIRSCHMULLER, H. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 30, n. 2, p. 328–341, 2008.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of stereo matching costs on images with radiometric differences. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 31, n. 9, p. 1582–1599, 2008.

HORNUNG, A.; WURM, K. M.; BENNEWITZ, M.; STACHNISS, C.; BURGARD, W. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, Springer, v. 34, n. 3, p. 189–206, 2013.

HUBER, P. J. Robust estimation of a location parameter. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 492–518.

ISHIMARU, P. J. A. *Desenvolvimento de módulo para segmentação de espaço livre em imagens estéreo com prototipação em FPGA*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2021.

IZADI, S.; KIM, D.; HILLIGES, O.; MOLYNEAUX, D.; NEWCOMBE, R.; KOHLI, P.; SHOTTON, J.; HODGES, S.; FREEMAN, D.; DAVISON, A. et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. [S.l.: s.n.], 2011. p. 559–568.

JASPERS, H.; WUENSCHE, H.-J. Fast and robust b-spline terrain estimation for off-road navigation with stereo vision. In: IEEE. *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. [S.l.], 2014. p. 140–145.

KHAN, M. S. *3D Robotic Mapping and Place Recognition*. Tese (Doutorado) — Technische Universität München, 2017.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In: IEEE. *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. [S.l.], 2007. p. 225–234.

KÜMMERLE, R.; GRISETTI, G.; STRASDAT, H.; KONOLIGE, K.; BURGARD, W. g 2 o: A general framework for graph optimization. In: IEEE. *2011 IEEE International Conference on Robotics and Automation*. [S.l.], 2011. p. 3607–3613.

KWEON, I.-S.; HEBERT, M.; KROTKOV, E.; KANADE, T. Terrain mapping for a roving planetary explorer. In: IEEE. *IEEE International Conference on Robotics and Automation*. [S.l.], 1989. p. 997–1002.

LABATUT, P.; PONS, J.-P.; KERIVEN, R. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: IEEE. *2007 IEEE 11th international conference on computer vision*. [S.l.], 2007. p. 1–8.

LABBE, M.; MICHAUD, F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, IEEE, v. 29, n. 3, p. 734–745, 2013.

LABBÉ, M.; MICHAUD, F. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, Wiley Online Library, v. 36, n. 2, p. 416–446, 2019.

LAVALLE, S. M. *Planning algorithms*. [S.l.]: Cambridge university press, 2006.

LAVALLE, S. M.; KUFFNER, J. J.; DONALD, B. et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, v. 5, p. 293–308, 2001.

LAZAROS, N.; SIRAKOULIS, G. C.; GASTERATOS, A. Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, Taylor & Francis, v. 2, n. 4, p. 435–462, 2008.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, Springer, v. 81, n. 2, p. 155, 2009.

LEUTENEGGER, S.; LYNEN, S.; BOSSE, M.; SIEGWART, R.; FURGALE, P. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 34, n. 3, p. 314–334, 2015.

LI, L.; LIU, Z.; ÖZGÏNER, Ü.; LIAN, J.; ZHOU, Y.; ZHAO, Y. Dense 3d semantic slam of traffic environment based on stereo vision. In: IEEE. *2018 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2018. p. 965–970.

LI, R.; WANG, S.; GU, D. Ongoing evolution of visual slam from geometry to deep learning: challenges and opportunities. *Cognitive Computation*, Springer, v. 10, n. 6, p. 875–889, 2018.

LI, Z.; ZHANG, X.; MÜLLER, H.; ZHANG, S. Large-scale retrieval for medical image analytics: A comprehensive review. *Medical image analysis*, Elsevier, v. 43, p. 66–84, 2018.

LING, Y. *Building maps for autonomous navigation using sparse visual SLAM features*. 2017. <https://www.youtube.com/watch?v=IrW9vFeaC7U>. [Online; Acessado 20-Setembro-2021].

LING, Y.; SHEN, S. Building maps for autonomous navigation using sparse visual slam features. In: IEEE. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2017. p. 1374–1381.

LING, Y.; SHEN, S. Real-time dense mapping for online processing and navigation. *Journal of Field Robotics*, Wiley Online Library, v. 36, n. 5, p. 1004–1036, 2019.

LLOYD, S. Least squares quantization in pcm. *IEEE transactions on information theory*, IEEE, v. 28, n. 2, p. 129–137, 1982.

LOPEZ, M.; MARI, R.; GARGALLO, P.; KUANG, Y.; GONZALEZ-JIMENEZ, J.; HARO, G. Deep single image camera calibration with radial distortion. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 11817–11825.

LOVI, D.; BIRKBECK, N.; COBZAS, D.; JAGERSAND, M. Incremental free-space carving for real-time 3d reconstruction. In: *Fifth international symposium on 3D data processing visualization and transmission (3DPVT)*. [S.l.: s.n.], 2010.

LOWRY, S.; SÜNDERHAUF, N.; NEWMAN, P.; LEONARD, J. J.; COX, D.; CORKE, P.; MILFORD, M. J. Visual place recognition: A survey. *IEEE Transactions on Robotics*, IEEE, v. 32, n. 1, p. 1–19, 2015.

LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, Springer, v. 4, n. 4, p. 333–349, 1997.

MACENSKI, S.; MARTÍN, F.; WHITE, R.; CLAVERO, J. G. The marathon 2: A navigation system. In: IEEE. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2020. p. 2718–2725.

MATHWORKS. *What Is Camera Calibration?* 2016. <https://de.mathworks.com/help/vision/ug/camera-calibration.html>. [Online; Acessado 20-Setembro-2021].

MEAGHER, D. Geometric modeling using octree encoding. *Computer graphics and image processing*, Elsevier, v. 19, n. 2, p. 129–147, 1982.

MELLO, R. S.; DORIA, N. S.; CONCEIÇAO, A. G.; FARIAS, P. C. Probabilistic roadmap qualitative study between two mapping methods rtab-map and hector slam applied to path planning. *III Brazilian Humanoid Robot Workshop and IV Brazilian Workshop on Service Robotics*, 2020.

MELO, M. *Occupancy Grid Map Estimation Based on Visual SLAM and Image Segmentation*. 2021. <https://www.youtube.com/watch?v=wLMdYD8zMy0>. [Online; Acessado 20-Setembro-2021].

MELO, M. S. P. de; NETO, J. G. da S.; SILVA, P. J. L. da; TEIXEIRA, J. M. X. N.; TEICHRIEB, V. Analysis and comparison of robotics 3d simulators. In: IEEE. *2019 21st Symposium on Virtual and Augmented Reality (SVR)*. [S.l.], 2019. p. 242–251.

MERZLYAKOV, A.; MACENSKI, S. A comparison of modern general-purpose visual slam approaches. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2021.

MILLER, J. N. Tutorial review—outliers in experimental data and their treatment. *Analyst*, The Royal Society of Chemistry, v. 118, n. 5, p. 455–461, 1993.

MORÉ, J. J. The levenberg-marquardt algorithm: implementation and theory. In: *Numerical analysis*. [S.l.]: Springer, 1978. p. 105–116.

MUNDY, J. L.; ZISSERMAN, A. Projective geometry for machine vision. Citeseer, 1992.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, IEEE, v. 31, n. 5, p. 1147–1163, 2015.

MUR-ARTAL, R.; TARDÓS, J. D. Fast relocalisation and loop closing in keyframe-based slam. In: IEEE. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2014. p. 846–853.

MUR-ARTAL, R.; TARDÓS, J. D. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In: ROME. *Robotics: Science and Systems*. [S.l.], 2015. v. 2015.

MUR-ARTAL, R.; TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, IEEE, v. 33, n. 5, p. 1255–1262, 2017.

MUR-ARTAL, R.; TARDÓS, J. D. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, IEEE, v. 2, n. 2, p. 796–803, 2017.

NISTER, D.; STEWENIUS, H. Scalable recognition with a vocabulary tree. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.], 2006. v. 2, p. 2161–2168.

OLEYNIKOVA, H.; MILLANE, A.; TAYLOR, Z.; GALCERAN, E.; NIETO, J.; SIEGWART, R. Signed distance fields: A natural representation for both mapping and planning. In: UNIVERSITY OF MICHIGAN. *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. [S.l.], 2016.

OLEYNIKOVA, H.; TAYLOR, Z.; FEHR, M.; SIEGWART, R.; NIETO, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In: IEEE. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2017. p. 1366–1373.

PELLENZ, J.; PAULUS, D. Mapping and map scoring at the robocuprescue competition. In: CITESEER. *Quantitative performance evaluation of navigation solutions for mobile robots (RSS 2008, Workshop CD)*. [S.l.], 2008. v. 46, p. 47–66.

PUMAROLA, A.; VAKHITOV, A.; AGUDO, A.; SANFELIU, A.; MORENO-NOGUER, F. Pl-slam: Real-time monocular visual slam with points and lines. In: IEEE. *2017 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2017. p. 4503–4508.

QI, X.; WANG, W.; YUAN, M.; WANG, Y.; LI, M.; XUE, L.; SUN, Y. Building semantic grid maps for domestic robot navigation. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 17, n. 1, p. 1729881419900066, 2020.

QINGGUO, T.; LONGSHENG, C. M-estimation and b-spline approximation for varying coefficient models with longitudinal data. *Journal of Nonparametric Statistics*, Taylor & Francis, v. 20, n. 7, p. 611–625, 2008.

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.

ROTH-TABAK, Y.; JAIN, R. Building an environment model using depth information. *Computer*, IEEE, v. 22, n. 6, p. 85–90, 1989.

ROUSSEEUW, P. J.; HUBERT, M. Robust statistics for outlier detection. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, Wiley Online Library, v. 1, n. 1, p. 73–79, 2011.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. Orb: An efficient alternative to sift or surf. In: IEEE. *2011 International conference on computer vision*. [S.l.], 2011. p. 2564–2571.

RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). In: IEEE. *2011 IEEE international conference on robotics and automation*. [S.l.], 2011. p. 1–4.

SAPUTRA, M. R. U.; MARKHAM, A.; TRIGONI, N. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 2, p. 1–36, 2018.

SCHARSTEIN, D.; SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, Springer, v. 47, n. 1, p. 7–42, 2002.

SCHAUWECKER, K.; ZELL, A. Robust and efficient volumetric occupancy mapping with an application to stereo vision. In: IEEE. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2014. p. 6102–6107.

SOILLE, P. Erosion and dilation. In: *Morphological Image Analysis*. [S.l.]: Springer, 2004. p. 63–103.

STUMBERG, L. von; USENKO, V.; ENGEL, J.; STÜCKLER, J.; CREMERS, D. Autonomous exploration with a low-cost quadrocopter using semi-dense monocular slam. *arXiv preprint arXiv:1609.07835*, 2016.

SUMIKURA, S.; SHIBUYA, M.; SAKURADA, K. Openvslam: A versatile visual slam framework. In: *Proceedings of the 27th ACM International Conference on Multimedia*. [S.l.: s.n.], 2019. p. 2292–2295.

SVOBODA, T.; MARTINEC, D.; PAJDLA, T. A convenient multicamera self-calibration for virtual environments. *Presence: Teleoperators & virtual environments*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 14, n. 4, p. 407–422, 2005.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2010.

TAKETOMI, T.; UCHIYAMA, H.; IKEDA, S. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, Springer, v. 9, n. 1, p. 16, 2017.

THRUN, S. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, Springer, v. 15, n. 2, p. 111–127, 2003.

TRIEBEL, R.; PFAFF, P.; BURGARD, W. Multi-level surface maps for outdoor terrain mapping and loop closing. In: IEEE. *2006 IEEE/RSJ international conference on intelligent robots and systems*. [S.l.], 2006. p. 2276–2282.

VIOLA, P.; III, W. M. W. Alignment by maximization of mutual information. *International journal of computer vision*, Springer, v. 24, n. 2, p. 137–154, 1997.

WANG, B.; WANG, H.; YU, Y.; ZONG, L. Orb-slam based semi-dense mapping with monocular camera. In: IEEE. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. [S.l.], 2017. p. 1–5.

WANG, S.; YUE, J.; DONG, Y.; SHEN, R.; ZHANG, X. Real-time omnidirectional visual slam with semi-dense mapping. In: IEEE. *2018 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2018. p. 695–700.

WEDEL, A.; BADINO, H.; RABE, C.; LOOSE, H.; FRANKE, U.; CREMERS, D. B-spline modeling of road surfaces with an application to free-space estimation. *IEEE transactions on Intelligent transportation systems*, IEEE, v. 10, n. 4, p. 572–583, 2009.

WERNER, D.; AL-HAMADI, A.; WERNER, P. Truncated signed distance function: experiments on voxel size. In: SPRINGER. *International Conference Image Analysis and Recognition*. [S.l.], 2014. p. 357–364.

WHELAN, T.; KAESS, M.; JOHANNSSON, H.; FALLON, M.; LEONARD, J. J.; MCDONALD, J. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, Sage Publications Sage UK: London, England, v. 34, n. 4-5, p. 598–626, 2015.

WHELAN, T.; LEUTENEGGER, S.; SALAS-MORENO, R.; GLOCKER, B.; DAVISON, A. Elasticfusion: Dense slam without a pose graph. In: ROBOTICS: SCIENCE AND SYSTEMS. [S.l.], 2015.

WHELAN, T.; SALAS-MORENO, R. F.; GLOCKER, B.; DAVISON, A. J.; LEUTENEGGER, S. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 35, n. 14, p. 1697–1716, 2016.

WILLIAMS, B.; CUMMINS, M.; NEIRA, J.; NEWMAN, P.; REID, I.; TARDÓS, J. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, Elsevier, v. 57, n. 12, p. 1188–1197, 2009.

XIA, L.; CUI, J.; SHEN, R.; XU, X.; GAO, Y.; LI, X. A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 17, n. 3, p. 1729881420919185, 2020.

XU, L.; FENG, C.; KAMAT, V. R.; MENASSA, C. C. An occupancy grid mapping enhanced visual slam for real-time locating applications in indoor gps-denied environments. *Automation in Construction*, Elsevier, v. 104, p. 230–245, 2019.

YAN, P.; LAN, Y.; YANG, S. A 3d grid mapping system based on depth prediction from a monocular camera. In: IEEE. *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. [S.l.], 2020. p. 564–570.

YAN, T.; GAN, Y.; XIA, Z.; ZHAO, Q. Segment-based disparity refinement with occlusion handling for stereo matching. *IEEE Transactions on Image Processing*, IEEE, v. 28, n. 8, p. 3885–3897, 2019.

YANG, S.; YANG, S.; YI, X.; YANG, W. Real-time globally consistent 3d grid mapping. In: IEEE. *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.], 2017. p. 929–935.

ZHANG, Y.; SU, Y.; YANG, J.; PONCE, J.; KONG, H. When dijkstra meets vanishing point: a stereo vision approach for road detection. *IEEE transactions on image processing*, IEEE, v. 27, n. 5, p. 2176–2188, 2018.

ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 22, n. 11, p. 1330–1334, 2000.

ZHAO, J.; KATUPITIYA, J.; WARD, J. Global correlation based ground plane estimation using v-disparity image. In: IEEE. *Proceedings 2007 IEEE international conference on robotics and automation*. [S.l.], 2007. p. 529–534.

ZOU, Q.; ZHU, Z. M-estimators for single-index model using b-spline. *Metrika*, Springer, v. 77, n. 2, p. 225–246, 2014.